Received: 00 Month 0000 | Revised: 00 Month 0000 | Accepted: 00 Month 0000

DOI: xxx/xxxx

ARTICLE TYPE

CUPS: Secure Opportunistic Cloud of Things Framework based on Attribute Based Encryption Scheme Supporting Access Policy Update

Sana Belguith*[1] | Nesrine Kaaniche[2] | Giovanni Russello[3]

[1] School of Computing, Science and Engineering, University of Salford, Manchester, UK
[2] Department of Computer Science, University of Sheffield, Sheffield, UK
[3] Cyber Security Foundry,University of Auckland, New Zealand, NZ

Correspondence
*Sana Belguith. Email: s.belguith@salford.ac.uk

Summary

The ever-growing number of Internet connected devices, coupled with the new computing trends, namely within emerging opportunistic networks, engenders several security concerns. Most of the exchanged data between the Internet of Things (IoT) devices are not adequately secured due to resource constraints on IoT devices. Attribute Based Encryption is a promising cryptographic mechanism suitable for distributed environments, providing flexible access control to encrypted data contents. However, it imposes high decryption costs, and does not support access policy update, for highly dynamic environments. This paper presents CUPS, an ABE-based framework for opportunistic cloud of things applications, that securely outsources data decryption process to edge nodes in order to reduce the computation overhead on the user side. CUPS allows end-users to offload most of the decryption overhead to an edge node and verify the correctness of the received partially decrypted data from the edge node. Moreover, CUPS provides the access policy update feature with neither involving a proxy-server, nor re-encrypting the enciphered data contents and re-distributing the users' secret keys. The access policy update feature in CUPS does not affect the size of the message received by the end-user which reduces the bandwidth and the storage usage. Our comprehensive theoretical analysis proves that CUPS outperforms existing schemes in terms of functionality, communication and computation overheads.

KEYWORDS:
Opportunistic computing, Cloud of things, Constant-size attribute based encryption, Decryption delegation, Verifiability, Access policy update, Confidentiality, Access control.

1 | INTRODUCTION

Opportunistic networks appeared as a promising type of Mobile Ad'hoc Networks, MANET for short[1]. Indeed, they enable direct communication between user-carried mobile devices, referred to as nodes, via short-range wireless technologies[2,3], instead of passing through the cellular network. Hence, opportunistic networks are considered as an interesting communication technology in several urban scenarios, e.g., in overloaded cellular network areas, and in non-urban scenarios, e.g., non-covered areas[4,5]. In a nutshell, opportunistic networks are defined as infrastructure-free: nodes store data, and carry it according to the related user mobility until a new communication opportunity arises to forward these data. This store-carry-forward feature was first introduced as a main defining function of Delay Tolerant Networks (DTN)[6]. Contrary to DTNs, in opportunistic networks, the focus shifts from user-oriented to content-oriented data dissemination. This considerably reduces the network complexity, as choosing appropriate intermediate nodes for forwarding data contents is no longer a priority. Instead, data dissemination depends on the mobility patterns of users as well as some shared contents' interests.

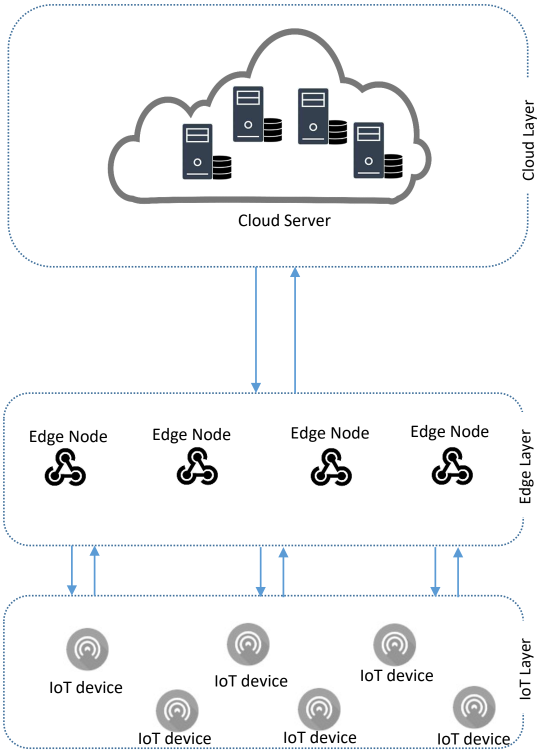

FIGURE 1 Cloud of Things Architecture

Recently, the Internet of Things (IoT) appeared and gained an expanding interest [7,8]. IoT has sparked a significant shift in all aspects of our lives including business, industry and society. Today's availability of low-cost embedded sensors and actuators offer unprecedented opportunity for interconnecting smart buildings, factories, vehicles, power grids and other data infrastructures [9,10,11,12]. Combining IoT and opportunistic networks has fostered a new computing paradigm referred to as Opportunistic Computing (*c.f.*, Figure 1). Exploiting devices' contact opportunities for communication is only a first step. Indeed, this device-to device communication can be used to share data, utilise other devices resources and compute tasks remotely.

Hence, adopting opportunistic computing to leverage secure content dissemination is an interesting solution, for IoT application [13,14]. However, in the context of opportunistic networks, secure content sharing remains an issue. An increasing need to design secure and efficient data sharing protocols adapted to IoT applications has appeared. Due to their limited storage and computation capacities, IoT devices are usually assisted with cloud services to store and process generated data [15,16,17], generally considered as a semi-trusted third party.

Data sharing schemes often rely on cryptographic algorithms to ensure the secrecy of shared data, between different group members. Although encryption mechanisms ensure data confidentiality against both curious cloud service providers and malicious users], the use of conventional encryption approaches is not sufficient to support the enforcement of fine-grained access control policies [15,18]. In fact, data confidentiality preservation becomes more complicated, considering flexible data sharing among highly dynamic group of users, trying to foster any opportunity to collect and/or retrieve data.

These opportunistic groups require efficient sharing of deciphering keys between different authorized users. In fact, the subscription of a new group member -based on its availability in an area-range and main interests- should not require updating the secret keys of existing users to minimise the key management complexity. Thus, the challenge lies in defining a comprehensive access control mechanism for outsourced data while both ensuring data secrecy and flexible up-datable access policies.

Attribute based Encryption (ABE) ensures encrypted access control to outsourced data while limiting privacy leakage of data producers and users [19,20,21]. ABE consists of enciphering data with respect to an access policy over a set of attributes where users that possess the matching attributes can recover data [8]. Nowadays, with the emergence of IoT applications adapted to distributed and dynamic environments, several settings require adding new users, strengthening access patterns and/or removing current users from systems. Hence, severe drawbacks that limit ABE application to resource constrained systems have to be considered. On the one hand, ABE techniques incur high decryption costs due to the execution of several pairing functions to verify and recover a plaintext. On the other hand, the update of access policies is not supported. Thus, the addition/revocation of users needs the re-encryption of the encrypted message and the re-distribution of users' secret keys.

Proxy re-encryption techniques are introduced and used to update users in outsourced systems. These mechanisms allow a server to re-encrypt stored data without accessing their content using a re-encryption key [22,23]. Attribute-based Proxy Re-encryption (AB-PRE) systems have been applied for access policy update. In AB-PRE, when the data owner decides to update access policies of some ciphertexts, she uses her private key to generated a re-encryption key for each ciphertext to be changed from an old access policy to a new one. Afterwards, all the generated re-encryption keys are uploaded to a proxy server to update the ciphertext using the received keys. Although AB-PRE schemes allow re-encryption without decrypting ciphertext or accessing the plaintexts, it requires that the data owner generates valid re-encryption keys. When the number of ciphertext rises, it becomes inefficient for a data owner to generate all the re-encryption keys and upload them to the proxy. Furthermore, this may also be unfeasible for limited bandwidths. Therefore, attribute-based proxy re-encryption schemes may not be efficient when updating a huge number of ciphertexts. Key Policy Attribute Based Encryption (KP-ABE) is widely applied to secure data in several distributed systems such as Publish and Subscribe systems (Pub/Sub) [24,25,26], pay-TV systems [27], vehicular networks [28], ..., where rules on who may read a document must be specified but it is unable to specify policies on a per-message basis. Obviously, these dynamic environments usually require efficiently adding new users and/or revoking existent users. That is, KP-ABE consists in labeling user's key by an access structure that specifies which type of ciphertext the key can decrypt, while ciphertext are labeled by a set of attributes. Thus, KP-ABE are adapted to distributed and decentralized environments. Instead, Ciphertext Policy Attribute Based Encryption (CP-ABE) schemes consists in associating an access structure to the ciphertext while assigning a set of credentials to deciphering users. CP-ABE schemes supporting policy update have been recently explored by Jiang et al. [29,30].

In this paper, we introduce CUPS, a key policy attribute based encryption scheme, that supports policy update and verifiable computation offloading, to leverage an opportunistic computing solution for cloud of things. CUPS is an extension of PU-ABE, our KP-ABE scheme supporting adding and/or removing attributes from the access policy without sharing keys with the cloud server [31]. PU-ABE [31], enables the encrypting entity to generate a ciphertext involving encrypted data together with some extra components used for supporting the access policy update feature. The ciphertext is forwarded to the cloud server that can update the access policy upon demand. Indeed, the cloud server does not need to be trusted. That is, it stores and shares ciphertexts among authorised users and also executes access policy update algorithm as requested. While executing these functionalities, the remote server is unable of decrypting any ciphertexts neither accessing any secret keys. In addition to this existent feature, this extension introduces several new functions introduced hereafter.

Contributions – CUPS presents a key policy attribute based encryption scheme which supports access policy update and verifiable decryption-outsourcing feature. This proposed scheme is suitable for bandwidth-limited applications as the size of the ciphertext received by end-users does not depend on the number of attributes involved in the access policy. In CUPS, the encrypting entity generates a ciphertext involving encrypted data together with some extra components used for supporting the access policy update feature. The ciphertext is forwarded to the cloud server that can update the access policy upon demand. Indeed, the cloud server does not need to be trusted. That is, it stores and shares ciphertexts among authorised users and also executes the access policy update algorithm as requested. While executing these functionalities, the remote server is unable of decrypting any ciphertexts neither accessing any secret keys. CUPS also supports secure delegation of the decryption algorithm to an edge node. This latter can partially decrypt the enciphered data content and forward the result to the requesting user. The user can then retrieve the plaintext by executing low cost mathematical operations. To achieve a secure delegation, CUPS allows the user to verify the accuracy of the received partially decrypted ciphertext to ensure that it has been honestly generated by the edge node. These features make CUPS is useful to be applied in dynamic environments requiring efficiently adding/removing users.

Our contributions are as follows:

1. CUPS provides a comprehensive framework for efficient and secure data sharing adapted to opportunistic cloud of things scenarios. It supports access policy update without requiring cipherext re-encryption or re-issuing users' secret keys. CUPS does not rely on a proxy re-encryption server to execute policy update procedures. In addition, unlike most ABE techniques, the size of the ciphertext that the end-user will receive is constant and independent from the number of attributes involved in the access policy. Therefore, CUPS generates a constant-size ciphertext, w.r.t. the end-user, that reduces bandwidth utilisation and storage costs.

2. CUPS extends the PU-ABE scheme [31] by adding a computation offloading feature. Indeed, the IoT device can offload to the nearest edge node a part of the decryption process. This latter performs most of the decryption operations without accessing the plaintext, and returns a partially decrypted ciphertext to the intended user. In return, the user can retrieve the plaintext by executing low cost mathematical operations.

3. CUPS allows the end-user to verify the accuracy of the partially decrypted message received from the edge node. This property is referred to as *verifiability*. Indeed, the user is able to check that the retrieved plaintext matches the ciphertext originally requested and downloaded from the cloud server.

Paper Organisation – The remainder of this work is as follows: Section 2 gives an overview of the proposed framework and describes the system and security models. Section 3 reviews related work and Section 4 presents the mathematical background. In Section 5, an overview of CUPS is introduced and the detailed construction is presented. Section 6 presents a rigorous security discussion. Finally, a theoretical analysis of computational performances is presented in Section 7, before concluding in Section 8.

2 | FRAMEWORK SPECIFICATION

In this section, we first present the network model, detailing the involved entities and their interactions in Section 2.1. Then, we detail the security requirements that the proposed system should fulfill in Section 2.2. Afterwards, we present the system and security models of the proposed CUPS framework in subsection 2.3 and subsection 2.5, respectively.

2.1 | Architecture

As presented in Figure 2 , CUPS framework considers a cloud service provider that stores data generated by data owners and share them among authorised users. Five different entities are defined as follows:

- The Central Trusted Authority (CTA), known by the Attribute authority, is responsible for generating the global public parameters and issuing users' secret keys. CTA is considered as a trusted entity in our model.
- The Cloud Server (RC) is a remote cloud server who stores and shares data among authorised users. RC is also responsible of executing the update algorithm to change the access policy involved in the ciphertext, w.r.t. the data owner's recommendations.
- Edge Node (EN) is responsible for partially decrypting a ciphertext using a transformation key received from the user. Indeed, the user derives a couple of public and private transformation keys from his secret keys. The user shares the public key with EN to allow the partial decryption of ciphertext, while keeping secret the private transformation key to be used for the final local data retrieval.
- The data owner (O) is the data producer. he defines access rights and encrypts data with respect to them before outsourcing to the cloud. In addition, the data owner generates extra ciphertext components used by the update algorithm.
- The data user (U) requests access to outsourced data. he delegates a part of the decryption process to EN. Specifically, U is responsible for using his secret keys to derive transformation keys used by EN to partially decrypt the ciphertext. Finally, U decrypts and verifies the partially decrypted ciphertext that is received from EN. A user may be malicious if he tries to access data without authorisation.

2.2 | Security Requirements

To design an efficient attribute based encryption scheme supporting efficient access policy update and computation delegation features, the following requirements need to be achieved:

- ***access policy update*** – our CUPS scheme should ensure adding new attributes and/or removing attributes from the access policy.
- ***flexible access control*** – our proposal should ensure flexible security policies among dynamic groups of users, w.r.t. forward secrecy and backward secrecy.
  - backward secrecy means that a new added user to a group is unable to decrypt information created prior to their introduction.
  - forward secrecy means that a compromise of the secret key does not affect the secrecy of future encrypted data.
- ***low computation overhead and storage cost*** – the proposed algorithms should have low processing complexity and acceptable storage cost to be adapted to resource-constrained devices and distributed environments.
- ***data confidentiality*** – our CUPS scheme has to protect the secrecy of outsourced and encrypted data contents against curious users and curious cloud service provider.

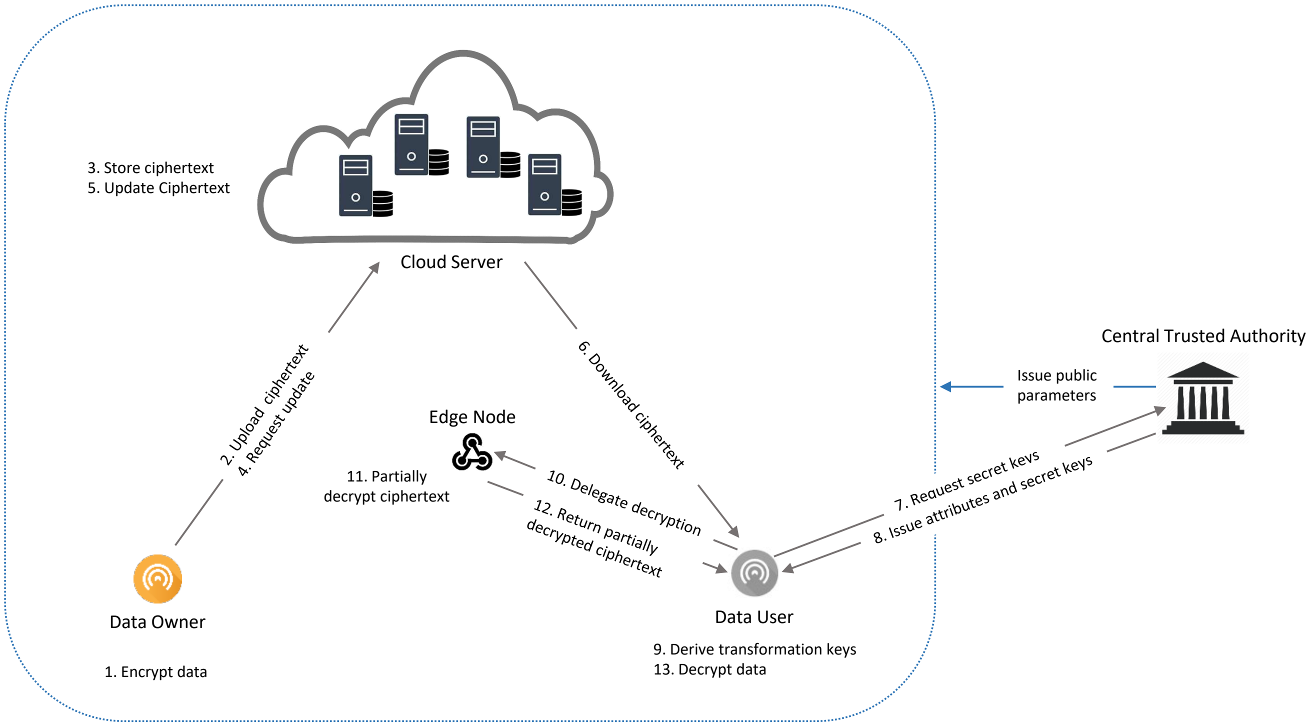

FIGURE 2 CUPS Architecture

2.3 | System Model

The CUPS framework is composed of four phases, i.e., SYS_INIT, STORAGE, UPDATE and RETRIEVAL, defined with respect to seven randomized algorithms detailed hereafter.

The SYS_INIT phase is executed once by the CTA. It permits to generate and publish system public parameters to *all* involved entities and derives users' private keys associated to their attributes. The SYS_INIT phase is based on two algorithms, denoted by setup and keygen.

During the STORAGE phase, the data owner (O) defines the access policy, i.e., set of attributes $\mathcal{S}$. This phase includes one randomized algorithm, denoted by encrypt, to encipher the data content w.r.t. the access policy $\mathcal{S}$, while pointing out either the content can be updated or not.

The UPDATE phase is executed by the cloud provider, upon the request of the data owner. It is based on one algorithm, denoted by update that supports both the addition and removal of attributes from access policies.

During the RETRIEVAL phase, the user (U) has first to request access to a particular data content, for the cloud provider. Once retrieved from RC, U runs an interactive protocol with the edge node (EN), to recover the original data content. The RETRIEVAL phase relies on three different algorithms, referred to as transform, $\mathsf{decrypt}_{\mathsf{out}}$ and decrypt. Indeed, U executes the transform algorithm to derive a *transformation* key, relying on his private keys that satisfy the encryption set of attributes $\mathcal{S}$ associated to the requested data content. The transformation key is then sent to EN along with the downloaded ciphertext from RC. This latter performs the $\mathsf{decrypt}_{\mathsf{out}}$ algorithm and generates a partially decrypted data content. Finally, based on the partially deciphered data file, U is able to finalise the decryption process. Recall that U is able check whether the received partially deciphered content was correctly generated, thanks to the support of the *verifiability* property.

The proposed CUPS framework consists of seven randomized algorithms: setup, encrypt, update, keygen, transform, $\mathsf{decrypt}_{\mathsf{out}}$ and decrypt, defined w.r.t. the four procedures, as follows:

- SYS_INIT phase:

  $\mathsf{setup}(\xi) \rightarrow (\mathtt{pp}, \mathtt{msk})$ – the setup algorithm is performed by a central trusted authority, known by the attribute authority. It takes as input a security parameter ξ and outputs the public parameters $\mathtt{pp}$ and the secret master key $\mathtt{msk}$.

  $\mathsf{keygen}(\mathtt{pp}, \mathtt{msk}, \Psi) \rightarrow \mathsf{sk}$ – this randomized algorithm is executed by the attribute authority to derive the secret keys of a user U. Given the public parameters $\mathtt{pp}$, an access policy Ψ of the user U and the secret master key $\mathtt{msk}$. The algorithm outputs the user's secret key sk w.r.t. to Ψ.

- STORAGE phase:
  $\mathsf{encrypt}(\mathtt{pp}, \mathcal{S}, \mathsf{M}) \rightarrow \mathsf{CT}$ – the encryption algorithm is performed by the data owner (O). It takes as inputs the public parameters $\mathtt{pp}$, the set of the encryption attributes $\mathcal{S}$ and the message M. This algorithm outputs the encrypted message, referred to as CT.

- UPDATE phase:
  $\mathsf{update}(\mathtt{pp}, \mathsf{CT}, \mathsf{ind}, \mathcal{U}) \rightarrow \mathsf{CT}'$ – the update algorithm is executed by a cloud server. It takes as inputs the public parameters $\mathtt{pp}$, a ciphertext CT that contains the set of enciphering attributes $\mathcal{S} = \{\mathsf{a}_i\}_{i=1..m}$ such that $|\mathcal{S}| = \mathsf{m}$, an operation indicator ind where ind = add or ind = revoke and a set of attributes $\mathcal{U}$ with $\mathcal{U} \cap \mathcal{S} = \emptyset$ if ind = add or $\mathcal{U} \subset \mathcal{S}$ if ind = revoke. It outputs a new ciphertext CT' for the new encrypting set of attributes $\mathcal{S}'$ such as $\mathcal{S}' = \mathcal{S} \cup \mathcal{U}$ or $\mathcal{S}' = \mathcal{S} \setminus \mathcal{U}$ w.r.t. ind value.

- RETRIEVAL phase:
  $\mathsf{transform}(\mathtt{pp}, \Psi, \mathsf{sk}) \rightarrow \mathtt{tk}$ – the transform algorithm is performed by U having an access policy Ψ and their related secret keys sk. transform takes as input $\mathtt{pp}$ and sk. It generates the transformation key $\mathtt{tk} = (\mathtt{tpk}, \mathtt{tsk})$ related to sk, where tpk and tsk are the public and private transformation keys, respectively.

  $\mathsf{decrypt}_{\mathsf{out}}(\mathtt{pp}, \mathsf{sk}, \mathsf{CT}') \rightarrow \mathsf{Y}$ – is executed by EN . To retrieve the partially decrypted message Y, this algorithm takes as input $\mathtt{pp}$, the transformation key tpk, a set of attributes Ψ and the updated ciphertext CT'. Note that CT' can be CT if the update algorithm has not been executed.

  $\mathsf{decrypt}(\mathsf{Y}, \mathtt{tsk}) \rightarrow \mathsf{M}$ –U executes decrypt to retrieve M. This algorithm takes tsk and the partially decrypted ciphertext Y as input and outputs M.

2.4 | Threat Model

In this section, we explain the considered attack model based on which we discuss the security properties of our proposed scheme.

- *An honest but curious cloud server provider (RC)*. RC is honest as it generates accurate inputs or outputs, during the different steps of the protocol, and performs calculations properly. However, it is curious to gain extra data from the protocol, such as obtaining credentials/attributes of a data user, retrieving the plaintext, or distinguishing the data owner based on the signcrypted content. As such, we consider the honest but curious attack model against the confidentiality w.r.t adaptive replayable chosen-plaintext attacks (IND-RCPA) (c.f, 2.5.1).

- *An unauthorised data user*. This attacker could be a data user (or an external entity), whose attributes do not satisfy the access policy associated with the ciphertext, or could be a revoked user. We also consider a set of colluding users on the attributes, who do not satisfy the access policy associated with the ciphertext and try to merge their attributes to retrieve the plaintext, in this attack model. These unauthorised users attempt to decrypt the content and access the plaintext. As such, we consider this attack model against the confidentiality requirements w.r.t adaptive replayable chosen-plaintext attacks (IND-RCPA) (c.f, 2.5.1).

- *A lazy or malicious edge node (EN)*. This attacker tries to forge a non-authentic partial decrypted ciphertext, or returns an old previously computed ciphertext. A security scheme against this attacker should satisfy verifiability feature (c.f, 2.5.2).

2.5 | Security Model

In this section, we explain the considered attack model based on which we discuss the security properties of our proposed scheme.

2.5.1 | Confidentiality

For designing a secure policy-update attribute based encryption scheme, we consider the case of malicious adversaries with respect to the indistinguishability property. The indistinguishability property means that if an adversary has some information about the plaintext, he should not learn about the ciphertext. This security notion requires the computational impossibility to distinguish between two messages chosen by the adversary with a probability greater than a half.

To design the most suitable security model considering the confidentiality requirement, we adopt an updated security model to capture security requirements related to policy updates [30]. CUPS is said to be indistinguishable against non-adaptive chosen ciphertext attacks if there is no probabilistic polynomial time (PPT) adversary that can win the Exp^{conf} security game with non-negligible advantage. The Exp^{conf} game is formally defined, between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows:

INITIALISATION – $\mathcal{A}$ selects a set of encryption attributes $\mathcal{S}^*$ (i.e; $\mathcal{S}^*$ corresponds to the set of attributes specified for the encryption) to be used for encrypting the challenge ciphertext, as a set of attributes $\mathcal{S}^* = \{\mathsf{a}_\mathsf{i}\}_{\mathsf{i}=1..\mathsf{m}}$ where $|\mathcal{S}^*| = \mathsf{m}$. $\mathcal{A}$ sends $\mathcal{S}^*$ to $\mathcal{C}$.

SETUP – the challenger $\mathcal{C}$ runs the $\mathsf{setup}(\xi)$ algorithm, sends the public parameters $\mathtt{pp}$ to the adversary $\mathcal{A}$ and keeps secret the master key $\mathtt{msk}$.

Phase 1 Queries – first, $\mathcal{C}$ sets an empty table T. Then, $\mathcal{A}$ is able to request the following queries, for each session i:

- **Private Key Query** – the adversary $\mathcal{A}$ queries an access policy $\Psi_{\mathcal{A},\mathsf{i}}$. The challenger $\mathcal{C}$ answers by running the $\mathsf{keygen}(\mathtt{pp}, \mathtt{msk}, \Psi_{\mathcal{A},\mathsf{i}})$ algorithm and sends the resulting secret key $\mathsf{sk}_{\mathcal{A},\mathsf{i}}$ to the adversary $\mathcal{A}$. Note that the access policy $\Psi_{\mathcal{A},\mathsf{i}}$ does not satisfy the encryption attribute set $\mathcal{S}^*$.

- **Transformation Key Query** –$\mathcal{A}$ queries the secret transformation keys $\mathsf{tk}_{\mathcal{A},\mathsf{i}}$, w.r.t. $\Psi_{\mathcal{A},\mathsf{i}}$. $\mathcal{C}$ searches the entry $(\Psi_{\mathcal{A},\mathsf{i}}, \mathsf{sk}_{\mathcal{A},\mathsf{i}}, \mathsf{tk}_{\mathcal{A},\mathsf{i}})$ in table T. It returns the transformation key if it exists in table T, otherwise $\mathcal{C}$ executes the $\mathsf{transform}$ algorithm to generate $\mathsf{tk}_{\mathcal{A},\mathsf{i}}$ and forwards them to the adversary.

CHALLENGE PHASE – during the challenge phase, $\mathcal{A}$ picks two equal length cleartexts $\mathsf{M}_0{}^*$ and $\mathsf{M}_1{}^*$ as well as an attribute set $\mathcal{U}^*$ with $|\mathcal{U}^*| = \mathsf{t}$ $\mathcal{U}^* \cap \mathcal{S}^* = \emptyset$ if ind = add or $\mathcal{U}^* \subset \mathcal{S}^*$ if ind = revoke. The challenger $\mathcal{C}$ chooses a random bit b from $\{0, 1\}$ with $\mathcal{S}'^* = \mathcal{S}^* \setminus \mathcal{U}^*$ for ind = add or $\mathcal{S}'^* = \mathcal{S}^* \cap \mathcal{U}^*$ for ind = revoke and computes the challenge encrypted message $\mathsf{CT}_\mathsf{b}{}^* = \mathsf{encrypt}(\mathtt{pp}, \mathcal{S}^*, \mathsf{M}_\mathsf{b}{}^*)$. It gives $\mathsf{CT}_\mathsf{b}{}^*$ to the adversary if $\mathcal{U} = \emptyset$, otherwise $\mathsf{CT}'^*_\mathsf{b} = \mathsf{update}(\mathtt{pp}, \mathsf{CT}_\mathsf{b}{}^*, \mathsf{ind}, \mathcal{U}^*)$.

QUERY PHASE 2 – in this phase, the adversary $\mathcal{A}$ can query a polynomially bounded number of queries as in QUERY PHASE 1, except that $\mathcal{A}$ cannot query secret keys related to an access policy Ψ that satisfies $\mathcal{S}^*$.

GUESS – $\mathcal{A}$ tries to guess which message M_i, where $\mathsf{i} \in \{0, 1\}$ corresponds to the enciphered data $\mathsf{CT}_\mathsf{b}{}^*$. Thus, $\mathcal{A}$ outputs a bit b' of b and wins the game if $\mathsf{b} = \mathsf{b}'$. The advantage of the adversary $\mathcal{A}$ in the above game is defined as:

$$Adv_{\mathcal{A}}[Exp^{Conf}(1^{\xi})] = |Pr[b = b'] - \frac{1}{2}|$$

Definition 1. CUPS fulfills the confidentiality property if there is no adversary that can succeed the security game $\mathsf{Exp}^{\mathsf{conf}}$ with non-negligible advantage.

2.5.2 | Verifiability

This scheme is said to be verifiable if there is no probabilistic polynomial time (PPT) adversary that can win the Exp^{verif} security game with non-negligible advantage. The Exp^{verif} game is formally defined, between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows:

INITIALISATION – $\mathcal{A}$ selects a set of encryption attributes $\mathcal{S}^*$ (i.e; $\mathcal{S}^*$ corresponds to the set of attributes specified for the encryption) to be used for encrypting the challenge ciphertext, as a set of attributes $\mathcal{S}^* = \{\mathsf{a}_\mathsf{i}\}_{\mathsf{i}=1..\mathsf{m}}$ where $|\mathcal{S}^*| = \mathsf{m}$. $\mathcal{A}$ sends $\mathcal{S}^*$ to $\mathcal{C}$.

SETUP – the challenger $\mathcal{C}$ runs the $\mathsf{setup}(\xi)$ algorithm, sends the public parameters $\mathtt{pp}$ to the adversary $\mathcal{A}$ and keeps secret the master key $\mathtt{msk}$.

Phase 1 Queries – first, $\mathcal{C}$ sets an empty table T. Then, $\mathcal{A}$ is able to request the following queries, for each session i:

- **Private Key Query** – the adversary $\mathcal{A}$ queries an access policy $\Psi_{\mathcal{A},\mathsf{i}}$. The challenger $\mathcal{C}$ answers by running the $\mathsf{keygen}(\mathtt{pp}, \mathtt{msk}, \Psi_{\mathcal{A},\mathsf{i}})$ algorithm and sends the resulting secret key $\mathsf{sk}_{\mathcal{A},\mathsf{i}}$ to the adversary $\mathcal{A}$. Note that the access policy $\Psi_{\mathcal{A},\mathsf{i}}$ does not satisfy the encryption attribute set $\mathcal{S}^*$.

- **Transformation Key Query** –$\mathcal{A}$ queries the secret transformation keys $\mathsf{tk}_{\mathcal{A},\mathsf{i}}$, w.r.t. $\Psi_{\mathcal{A},\mathsf{i}}$. $\mathcal{C}$ searches the entry $(\Psi_{\mathcal{A},\mathsf{i}}, \mathsf{sk}_{\mathcal{A},\mathsf{i}}, \mathsf{tk}_{\mathcal{A},\mathsf{i}})$ in table T. It returns the transformation key if it exists in table T, otherwise $\mathcal{C}$ executes the $\mathsf{transform}$ algorithm to generate $\mathsf{tk}_{\mathcal{A},\mathsf{i}}$ and forwards them to the adversary.

Challenge – during the challenge phase, $\mathcal{A}$ chooses a challenge message M^* and sends it to the challenger. The challenger C encrypts M^* and generates the verification key V^* under $\mathcal{S}^*$. Then, the generated ciphertext CT^* is returned to the adversary.

QUERY PHASE 2 – in this phase, the adversary $\mathcal{A}$ can query a polynomially bounded number of queries as in QUERY PHASE 1, except that $\mathcal{A}$ cannot query secret keys related to an access policy Ψ that satisfies $\mathcal{S}^*$.

Forge – $\mathcal{A}$ generates a random partially decrypted ciphertext Y^* without executing the $\mathsf{Decrypt}_{\mathsf{out}}$ algorithm. $\mathcal{A}$ wins the game if $\mathsf{Decrypt}(\mathtt{pp}, \mathsf{tpk}, \mathcal{S}^*, \mathsf{Y}^*) \notin \{\mathsf{M}^*, \perp\}$ and the verification of the partially decrypted ciphertext is valid.

$\mathcal{A}$'s advantage is noted as:

$$Adv_{\mathcal{A}}[Exp^{verif}(1^{\xi})] = |Pr[Exp^{verif}(1^{\xi})] = 1|$$

Definition 2. CUPS fulfills the verifiability property if there is no adversary that can succeed the security game Exp^{verif} with non-negligible advantage.

3 | ATTRIBUTE BASED ENCRYPTION SCHEMES

Attribute-based Encryption (ABE) has been designed to ensure encrypted flexible access control for outsourced data [32]. Unlike traditional public key encryption schemes, ABE consists in encrypting data for many users. Therefore, decrypting entities' private keys and encrypted data are labeled with a set of attributes or a structure over attributes. A user is able to decrypt a ciphertext if there is a match between her private key and the ciphertext [33]. Attribute based encryption schemes are classified into two categories, namely: Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE) [34]. In KP-ABE, ciphertexts are labeled with a set of attributes while users' private keys are associated with an access policy which can be any monotonic structure. The user is able to decrypt the ciphertext if her access policy is satisfied by the attributes embedded in the ciphertext. KP-ABE schemes have been widely applied to secure data in distributed systems such as Internet of Things, publish and subscribe systems, intelligent transport systems, etc. [35].

3.1 | Constant size KP-ABE

Although ABE schemes ensure flexible access control to encrypted data, the communication and computation overhead as well as the bandwidth consumption increase exponentially with the number of attributes required in the access policies. To save the storage cost of ciphertext and processing overhead of encryption, attribute based encryption schemes with constant ciphertext size have been introduced [36,37,38,39]. In these schemes, the size of the generated ciphertext does not depend on the number of attributes used on the threshold access policies, which presents an interesting feature mainly for resource-constrained devices. Herranz et al. [36] have proposed the first constant size threshold ciphertext-policy attribute based encryption scheme. Indeed, the ciphertext size is constant and does not depend on the number of attributes involved in the threshold access policies. Afterwards, several CP-ABE schemes with constant cipheretxt size have been proposed [37,40,41]. Due to the construction of CP-ABE schemes, monotone access policies based schemes can not be extended to ensure a constant ciphertext size. For instance, these schemes consist in only using threshold or conjunctive access policies which do not provide the desired expressiveness [42].

Several expressive KP-ABE schemes with constant ciphertext size have been designed [38,39,43]. Wang et al. [39] have proposed a KP-ABE scheme with constant ciphertext size. This scheme relies on a monotone access policy to express the users' attributes.

Although, these schemes ensure reduced communication and computation costs, they still present a major limitation which is their incapacity of changing access policies of ciphertexts. In dynamic environments, users may be often added or removed, then access policies should be updated to support these changes. Recently, the first CP-ABE with policy update has been proposed by Jiang et al. [29,30]. The authors introduced a new variant of CP-ABE supporting access policy update that captures the functionalities of attribute addition and revocation from access policies. They provide two CP-ABE schemes supporting AND-gate access policies with constant-size ciphertexts. The first KP-ABE scheme supporting attributes addition is recently introduced by Belguith et al. [44]. In this scheme, the ciphertext is updated by the cloud provider to add new attributes to the encryption set of attributes. Therefore, new users can be added and are able to decrypt the ciphertext without re-encrypting the ciphertext neither re-issuing existent users secret keys. Later, the same authors have extended their scheme to support attributes' revocation. The introduced scheme PU-ABE [31] support both attributes revocation and addition. Indeed, users can be added or removed from system without relying on a proxy server neither re-issuing secret keys nor re-encryption the ciphertext.

3.2 | Decryption Delegation in ABE

One key limitation of ABE schemes is the high decryption cost, which grows with the complexity of the access policies. Indeed, the decryption cost is related to the execution of several pairing functions [12,45,46,41,47].

To mitigate this issue, Green et al. [47] proposed to outsource the execution of the decryption algorithm to a semi trusted server. To this end, a user needs to derive a pair of new keys called public and private transformation keys based on his own secret key. Then, the cipheretext and the public transformation key are forwarded to the semi trusted server that is able to partially decrypt the ciphertext using the received transformation key without accessing the plaintext. This partially decrypted ciphertext is then returned to the user who uses the private transformation key to fully decrypt the ciphertext by performing only one exponentiation operation. By applying this technique, users reduce the computation costs at their side. The delegation of the decryption algorithm to semi trusted server requires that the user verifies the correctness of the received partially decrypted ciphertext. A lazy server may try to forward previously partially decrypted ciphertext [45]. To overcome this limitation, several verifiable outsourced ABE schemes were proposed [48,49].

Lai et al.[46] proposed an outsourced ABE scheme where the user is able to verify the correctness of the partially decrypted ciphertext. This scheme encrypts two different messages, one is the original message and the other is a random message. The ciphertext involves a component $\tilde{C}$ generated using a combined hash functions over both messages. After receiving the partially decrypted ciphertext, the user decrypts the two messages and compare the result to $\tilde{C}$ to verify the correctness of the decryption performed by the server.

An ABE scheme with both outsourced encryption and decryption algorithms was described by Wang et al.[50]. In this scheme, the data owner is assisted by a proxy server to partially encrypt the data. Then, the data owner uses the partially encrypted data to encrypt the message and generate the ciphertext to be outsourced to the cloud server. On the other side, users outsource the decryption algorithm to a semi-trusted server who partially decrypts the ciphertext. In 2017, Li et al.[41] proposed a CP-ABE scheme that supports the verifiable outsourced decryption feature. Beyond reducing decryption overhead, this scheme incurs low communication costs as it generates a constant-size ciphertext. Recently, PHOABE, the first multi authority CP-ABE scheme with outsourced decryption has been introduced[12]. In this scheme, a user may delegate the decryption algorithm to be executed by a semi-trusted server while being able to verify the correctness of the partially decrypted ciphertext.

4 | MATHEMATICAL BACKGROUND

In this section, we first introduce the access structure in section 4.1. Then, in section 4.2, we present the bilinear maps. Finally, we introduce some security assumptions.

4.1 | Access Policies

Access policies can be represented as a boolean functions of attributes or a Linear Secret Sharing Scheme ($\mathtt{LSSS}$) matrix.

Access Structure – Let $\mathcal{P} = \{\mathcal{P}_1, \cdots, \mathcal{P}_n\}$ be a set of parties. A collection $\mathtt{A} \subseteq 2^{\{\mathcal{P}_1,\cdots,\mathcal{P}_n\}}$ is monotone if $\forall B, C$ if $B \in \mathtt{A}$ and $B \subseteq C$ then $C \in \mathtt{A}$. An access structure is a collection $\mathtt{A}$ of non-empty subsets of $\{\mathcal{P}_1, \cdots, \mathcal{P}_n\}$, such as $\mathtt{A} \subseteq 2^{\{\mathcal{P}_1,\cdots,\mathcal{P}_n\}} \backslash \emptyset$. Note that any access structure can be converted into a boolean function. Boolean functions can be defined as an access tree, where the leaves present the attributes while the intermediate and the root nodes are the logical operators AND ($\wedge$) and OR ($\vee$).

Linear Secret Sharing Schemes ($\mathtt{LSSS}$***) –*** Let $\mathcal{P}$ be a set of parties, A be $l \times n$ matrix, and $\rho : \{1, 2, \cdots, l\} \to \mathcal{P}$ be a function that maps a row to a party for labeling. A secret sharing scheme for access structure Ψ over a set of parties $\mathcal{P}$ is a linear secret sharing scheme ($\mathtt{LSSS}$) in $\mathbb{Z}_p$ and is represented by (A, ρ) if it consists of two efficient algorithms:

- $\mathsf{Share}((A, \rho), s)$: The share algorithm takes as input $s \in \mathbb{Z}_p$ which is to be shared. It randomly chooses $\beta_1, \cdots, \beta_n \in \mathbb{Z}_p$, and defines $\beta = (\beta_1 = s, \beta_2, \cdots, \beta_k)^T$. It outputs $M.\beta$ as the vectors of l shares. The share $\lambda_i =< A_i, \beta^T >$ belongs to party $\rho(i)$, where A_i is the $i - \mathrm{th}$ of A.
- $\mathsf{Recon}((A, \rho), \mathcal{S})$: The reconstruction algorithm takes as input an access set $\mathcal{S} \in A$. Let $I = \{i | \rho(i) \in \mathcal{S}\}$. It outputs a set of constants $\{\mu_i\}_{i \in I}$ such that $\sum_{i \in I} \mu_i \cdot \lambda_i = \beta_1 = s$.

4.2 | Bilinear Maps

Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be three multiplicative groups of a finite field having the same order p. An admissible asymmetric pairing function $\hat{e}$ from $\mathbb{G}_1 \times \mathbb{G}_2$ in $\mathbb{G}_T$ has to be bilinear, non degenerate and efficiently computable.

4.3 | The General Diffie-Hellman Exponent Assumption

In our CUPS construction, we make use of the generalisation of the Diffie-Hellman exponent assumption, formally defined by Boneh et al. in[51]. The authors have introduced a class of assumptions that appeared with the use of pairing-based schemes namely Decisional DiffieâĂŞHellman assumption (DDH), Bilinear Diffie-Hellman (BDH), and q-Bilinear Diffie Hellman Exponent (qâĹŠBDHE) assumptions, detailed hereafter.

Let $B = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}, \hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G})$ be a bilinear map group such that $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. Let g_0 be a generator of $\mathbb{G}$ and set $g = \hat{e}(g_0, g_0) \in \mathbb{G}$. Let s and n be positive integers and $P, Q \in \mathbb{F}_p[X_1, \cdots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$ where $P = (p_1, \cdots, p_s)$ and $Q = (q_1, \cdots, q_s)$ and $p_1 = q_1 = 1$. For any function $h : \mathbb{F}_p \to \Omega$ and any vector $(x_1, \cdots, x_n) \in \mathbb{F}_p^n$, $h(P(x_1, \cdots, x_n))$

stands for $(h(p_1(x_1,\cdots,x_n)),\cdots,h(p_s(x_1,\cdots,x_n)))\in\Omega^s$ and $h(Q(x_1,\cdots,x_n))$ stands for $(h(q_1(x_1,\cdots,x_n)),\cdots,h(q_s(x_1,\cdots,x_n)))\in\Omega^s$. Let $f\in\mathbb{F}_p[X_1,\cdots,X_n]$, it is said that f depends on (P,Q), which we denote by $f\in P,Q$, when there is a linear decomposition

$$f=\sum_{1\leq i,j\leq s}a_{i,j}\cdot p_i\cdot p_j+\sum_{1\leq i\leq s}b_i\cdot q_I,\quad a_{i,j},b_i\in\mathbb{Z}_p$$

Let P, Q be as above and $f\in\mathbb{F}_p[X_1,\cdots,X_n]$. The (P,Q,f)-General Diffie- Hellman Exponent problems are defined as follows.

Definition 1: (P,Q,f)-***GDHE***. Given a tuple $H(x_1,\cdots,x_n)=(g_0^{P(x_1,\cdots,x_n)},g^{Q(x_1,\cdots,x_n)})\in\mathbb{G}_1^s\times\mathbb{G}^s$, compute $g^{f(x_1,\cdots,x_n)}$.

Definition 2: (P,Q,f)-***GDDHE***. Given $H(x_1,\cdots,x_n)\in\mathbb{G}_1^s\times\mathbb{G}^s$, compute $g^{f(x_1,\cdots,x_n)}$ as above, decide whether $T=g^{f(x_1,\cdots,x_n)}$.

We refer to [51] for a proof that (P,Q,f)-***GDHE*** and (P,Q,f)-***GDDHE*** have generic security when $f\notin<P,Q>$.

4.4 | Collision-Resistant Hash Functions

The proposed CUPS scheme relies on the use of collision-resistant hash functions, defined as follows:

Definition 3. Collision-Resistant Hash Function – A hash function $\mathcal{H}:\{0,1\}^n\longrightarrow\{0,1\}^m$, where $n,m\in\mathbb{N}$, is said to be collision-resistant if it satisfies the following two properties:

- *length compressing* — $m>n$, typically $m=n/2$;
- *hard to find collisions* — for all non-uniform probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, there exists a negligible function ϵ, such that for all $n\in\mathbb{N}$,

$$Pr[(x_0,x_1)\leftarrow\mathcal{A}(1^n,\mathcal{H}):x_0\neq x_1\wedge\mathcal{H}(x_0)=\mathcal{H}(x_1)]\leq\epsilon(n)$$

5 | CUPS: SECURE OPPORTUNISTIC CLOUD OF THINGS FRAMEWORK BASED ON ATTRIBUTE BASED ENCRYPTION SCHEME

In this section, we first give an overview of our proposed constant size CUPS scheme, while presenting the main procedures (subsection 5.1). Then, we detail the scheme construction, in subsection 5.2.

5.1 | Overview

CUPS presents a verifiable and outsourced KP-ABE scheme that ensures flexible access control, while supporting policy updates in opportunistic cloud of things environments. CUPS scheme relies on the constant size KP-ABE scheme proposed by Belguith et al.[31], which has been extended to fulfill the decryption delegation feature.

Figure 3 presents a detailed workflow of CUPS, while enhancing the different interactions between involved actors. Based on four phases, Figure 3 shows the chronological sequence of seven randomized algorithms and a set of functions. Recall that some phases such as the SYS_INIT, STORAGE and RETRIEVAL are compulsory, while the UPDATE phase is considered as optional. Similarly, some functions are considered as internal features such as the *verify* function that enables a honest user to find out whether a data content that has been decrypted with the edge node assistance is matching the ciphertext downloaded from remote server or not.

For ease of presentation, the different notations used in this paper are listed in Table 1 .

5.2 | Secure Opportunistic Cloud of Things: Proposed Protocol

Our CUPS construction, supporting both attributes' addition and revocation, is based on four phases including five algorithms defined as follows:

- SYS_INIT phase:
  This phase is executed once by the CTA. It permits to generate and publish system public parameters to *all* involved entities and derives users' private keys associated to their attributes relying on the following two algorithms:

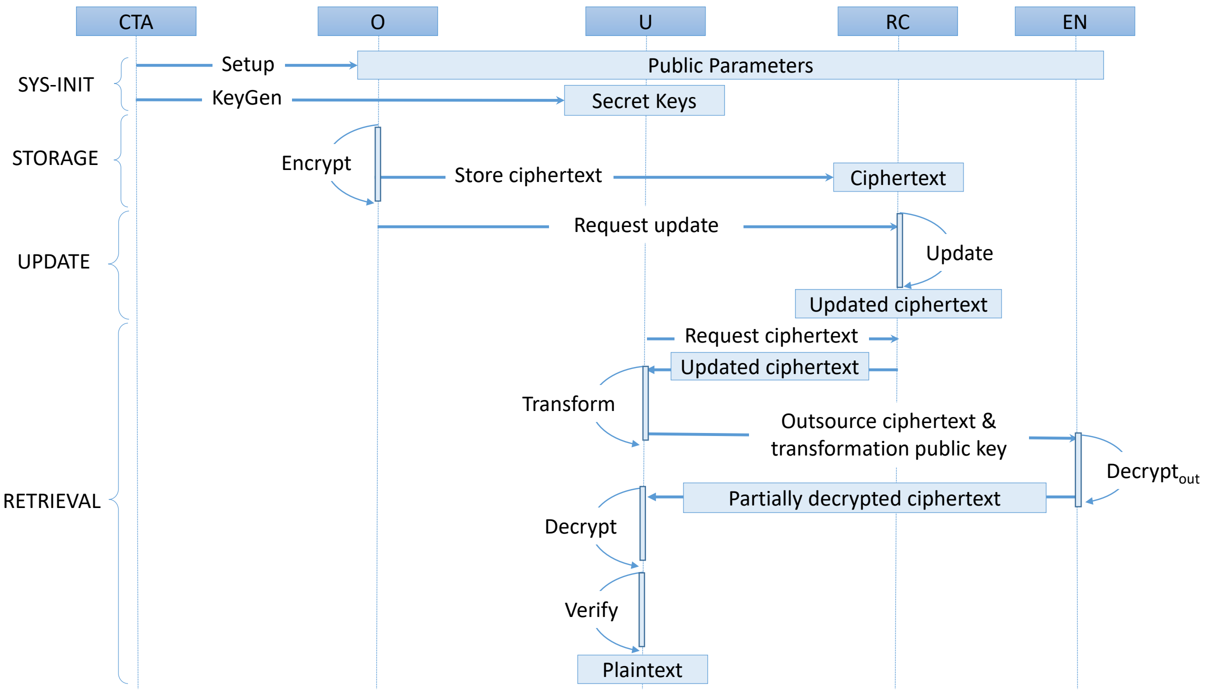

FIGURE 3 Work Flow of CUPS Protocol

- setup – given the security parameter ξ, the attribute authority chooses three cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p and defines a bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ and four collusion resistant hash functions $\mathcal{H}' : \{0,1\}^* \rightarrow \mathbb{Z}_P, \mathcal{H}_0 : \{\mathbb{M}\} \rightarrow \{0,1\}^{n_{\mathcal{H}_0}}, \mathcal{H}_1 : \{\mathbb{M}\} \rightarrow \{0,1\}^*, \mathcal{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^{n_{\mathcal{H}_2}}$, where $\mathbb{M}$ is the message universe and $n_{\mathcal{H}_0}$ and $n_{\mathcal{H}_2}$ are the output-sizes of H_0 and H_2 hash functions, respectively.. It also randomly selects two generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$ as well as a secret random value $\alpha \in \mathbb{Z}_p^*$. In addition, the attribute authority sets $v = \hat{e}(g,h)$, $\{h^{\alpha^i}\}_{i=1\cdots k}$ and $\{u_i = g^{\alpha^i}\}_{i=1\cdots k}$ where $k = |\mathbb{U}|$ is the cardinal of the attributes universes $\mathbb{U}$. Finally, it chooses a cryptographic hash function $\mathcal{H} : \{0,1\}^* \Rightarrow \mathbb{Z}_p^*$ and outputs the public parameters pp as follows:

$$\mathrm{pp} \ = \ (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, v = \hat{e}(g,h), h, \{h^{\alpha^i}\}_{i=1\cdots k}, \{u_i = g^{\alpha^i}\}_{i=1\cdots k})$$

  The master secret key is defined as $\mathrm{msk} = (g, \alpha)$.

- keygen – it computes the private key associated to an access structure Ψ w.r.t. an LSSS scheme (A, ρ) such that A is the corresponding $l \times n$ matrix. First, the keygen algorithm generates shares of 1 relying on the LSSS schema w.r.t. (A, ρ), as detailed in section 4. Namely, it chooses a column vector $\beta = (\beta_1, \beta_2, \cdots, \beta_n)^T$, while $\beta_1 = s = 1$ and $\beta_2, \cdots, \beta_n \in \mathbb{Z}_p$. Then for each $i = 1$ to l, it calculates $\lambda_i =< A_i, \beta^T >$, and sets sk as follows:

$$\begin{aligned} sk \ &= \ \{D_i\}_{i=0}^{l}, \{(K_{i,j})_{j=0}^{n}\}_{i=0}^{l} \\ &= \ \{g^{\frac{\lambda_i}{\alpha+\mathcal{H}(\rho(i))}}, (h^{\lambda_i \alpha^j})_{j=0}^{n}\}_{i=0}^{l} \end{aligned}$$

- STORAGE phase:

  - encrypt – let $\mathcal{S}$ be the set of the encryption attributes $\mathcal{S} = \{a_i\}_{i=1}^{m}$ and R a random value in $\mathbb{G}_T$. This algorithm, executed by the encrypting entity $\mathcal{E}$ takes an extra input which is a maximum revocation number $r \leq m$. This algorithms runs w.r.t following two steps: (i) The data owner chooses $s \in \mathbb{Z}_p^*$ and computes the ciphertext CT as follows:

TABLE 1 The different notations used in this paper

Notation	Description
RC	Remote Cloud Server
CTA	Central Trusted Authority
EN	Edge Node
O	Data Owner
U	User
M	Message
$\mathbb{U}$	The attribute universe
k	The size of the attribute universe $\mathbb{U}$
$\mathbb{M}$	The message universe
$\mathtt{pp}$	Public Parameters
$\mathtt{msk}$	Master Secret Key
a	An attribute
Ψ	A user access policy
$\mathcal{U}$	A set of attributes to be added/removed to an encrypting access policy
l	The size of $\mathcal{U}$
$\mathcal{S}$	An encrypting access policy
$\mathcal{S}'$	An updated encrypting access policy
sk	Secret key related to a user U
tk	A transformation key related to user U
tpk	A transformation public key
tsk	A transformation private key
CT	The ciphertext
CT′	The updated ciphertext
Y	The partially decrypted message
E_1	An exponentiation overhead in $\mathbb{G}_1$
E_2	An exponentiation overhead in $\mathbb{G}_2$
E	An exponentiation overhead in $\mathbb{G}$
τ_P	The computation overhead of a pairing function ê
$\mathcal{O}(M)$	The size of a message M
$\mathcal{H}$	The overhead of a hash function
m	The size of encrypting set of attributes or the encryption access policy
n	The size of the user access policy or the set of attributes used to generate his secret keys
r	The maximum number of attributes that can be revoked from an access policy
Ω	The size of a ciphertext element in bits
Φ	The size of a user's secret key element in bits

$$\begin{cases} E_0 = h^{s \cdot \prod_{a_i \in \mathcal{S}} (\alpha + \mathcal{H}(a_i))} \\ E_1 = E_0^{\alpha}, \cdots, E_{k-m} = E_{k-m-1}^{\alpha} \\ C_1 = u_1^{-s} =, \cdots, C_{r+1} = u_{r+1}^{-s} \\ C = R \cdot \hat{e}(g, h)^s \end{cases}$$

(ii) Afterwards, the algorithm computes $R_0 = \mathcal{H}_0(R)$ and a symmetric key $K = \mathcal{H}_1(R)$. O encrypts the message M using a symmetric encryption algorithm enc_K such that $C_K = \mathsf{enc}_K(K, M)$. The verification key is computed as $V = \mathcal{H}_2(R_0 || C_K)$.

- UPDATE phase:

  – $\mathtt{update}$ – the $\mathtt{update}$ algorithm first checks the operation indicator $\mathtt{ind}$. Then, if $\mathtt{ind}$ = $\mathtt{add}$, it proceeds as (i), otherwise if $\mathtt{ind}$ = $\mathtt{revoke}$ it executes (ii):

  (i) – given a ciphertext CT encrypted w.r.t. a set of attributes $\mathcal{S}$ and $\mathcal{U} = \{a'_1, \cdots, a'_t\}$ a new set of attributes where $\mathcal{U} \cap \mathcal{S} = \emptyset$, the server has to add elements of $\mathcal{U}$ to the set of encrypting attributes $\mathcal{S}$ of the ciphertext CT. To do so, it proceeds as follows:
  Let $F(x)$ be the polynomial in x defined as $F(x) = \prod_{a_i \in \mathcal{U}} (x + \mathcal{H}(a_i)) = f_t x^t + f_{t-1} x^{t-1} + \cdots + f_0$
  Then, the algorithm computes $E'_0 = E_0^{F(\alpha)} = \prod_{i=0}^{t} E_i^{f_i}$. The new ciphertext is then defined as $CT' = (E'_0, C_1, C)$ w.r.t. $\mathcal{S}'$, the new set of encrypting attributes defined as $\mathcal{S}' = \mathcal{S} \cup \mathcal{U}$.

  (ii) – given a ciphertext CT encrypted w.r.t. a set of attributes $\mathcal{S}$ and a revocation attribute set $\mathcal{U} = \{a'_1, \cdots, a'_t\} \subseteq \mathcal{S}$ where $t \leq r$, the server updates the ciphertext CT as follows:

Let $\mathsf{F}(\mathsf{x})$ be the polynomial in x as $\mathsf{F}(\mathsf{x}) = \frac{1}{\prod_{\mathsf{a}_\mathsf{i} \in \mathsf{U}} \mathcal{H}(\mathsf{a}_\mathsf{i})} \prod_{\mathsf{a}_\mathsf{i} \in \mathcal{U}} (\mathsf{x} + \mathcal{H}(\mathsf{a}_\mathsf{i})) = \mathsf{f}_\mathsf{t}\mathsf{x}^\mathsf{t} + \mathsf{f}_{\mathsf{t}-1}\mathsf{x}^{\mathsf{t}-1} + \cdots + \mathsf{f}_0$

Then, the algorithm computes CT' as follows:

$$\begin{cases} \mathsf{E}_0' = \mathsf{E}_0^{\frac{1}{\prod_{\mathsf{a}_\mathsf{i} \in \mathcal{U}} \mathcal{H}(\mathsf{a}_\mathsf{i})}} = \mathsf{h}^{\mathsf{s} \cdot \prod_{\mathsf{a}_\mathsf{i} \in \mathcal{S} \setminus \mathcal{U}} (\alpha + \mathcal{H}(\mathsf{a}_\mathsf{i})) \mathsf{F}(\alpha)} \\ \mathsf{C}_1' = \prod_{\mathsf{t}=1}^{\mathsf{t}+1} \mathsf{C}_\mathsf{i}^{\mathsf{f}_{\mathsf{i}-1}} = \mathsf{g}^{-\alpha \mathsf{s} \sum_{\mathsf{i}=1}^{\mathsf{t}+1} \alpha^{\mathsf{i}-1} \mathsf{f}_{\mathsf{i}-1}} = \mathsf{u}_1^{-\mathsf{s}\mathsf{F}(\alpha)} \\ \mathsf{C}' = \mathsf{C}\hat{\mathsf{e}}(\prod_{\mathsf{i}=1}^{\mathsf{t}} \mathsf{C}_\mathsf{i}^{-\mathsf{f}_\mathsf{i}}, \mathsf{h}) = \mathsf{R}\hat{\mathsf{e}}(\mathsf{g}, \mathsf{h})^{\mathsf{s} \sum_{\mathsf{i}=0}^{\mathsf{t}} \mathsf{f}_\mathsf{i} \alpha^\mathsf{i}} \\ = \mathsf{R}\mathsf{v}^{\mathsf{s}\mathsf{F}(\alpha)} \end{cases}$$

- RETRIEVAL phase:
  During the RETRIEVAL phase, U has first to request access to a particular data content, for the cloud provider. Once retrieved from RC, U runs an interactive protocol with the edge node (EN), to recover the original data content. The RETRIEVAL phase relies on the following three algorithms:

  - transform – U executes this algorithm to generate the transformation keys. He first picks a random value $\mathsf{z} \in \mathbb{Z}_\mathsf{N}^*$, then derives the transformation public and private keys $\mathtt{tk} = (\mathtt{tpk}, \mathtt{tsk})$, where $\mathtt{tpk}$ and $\mathtt{tsk}$ are computed as follows:

    $$\begin{cases} \mathtt{tpk} = \mathtt{sk}^{\frac{1}{z}} = (\{\mathsf{D}_\mathsf{i}\}_{\mathsf{i}=0}^{\mathsf{l}})^{\frac{1}{z}}, (\{(\mathsf{K}_{\mathsf{i},\mathsf{j}})_{\mathsf{j}=0}^{\mathsf{n}}\}_{\mathsf{i}=0}^{\mathsf{l}})^{\frac{1}{z}} \\ \\ \mathtt{tsk} = \mathsf{z} \end{cases}$$

    Therefore, $\mathtt{tpk}$ and $\mathtt{tsk}$ are defined as:

    $$\begin{cases} \mathtt{tpk} = \mathtt{sk}^{\frac{1}{z}} = \mathsf{g}^{\frac{\lambda_\mathsf{i}}{\mathsf{z}(\alpha + \mathcal{H}(\rho(\mathsf{i})))}}, (\mathsf{h}^{\frac{\lambda_\mathsf{i} \alpha^\mathsf{j}}{\mathsf{z}}})_{\mathsf{j}=0}^{\mathsf{n}}\}_{\mathsf{i}=0}^{\mathsf{l}} \\ \\ \mathtt{tsk} = \mathsf{z} \end{cases}$$

    Finally, U outsources the ciphertext as downloaded from RC along with the transformation public key $\mathtt{tpk}$ to the EN.

  - $\mathsf{decrypt}_{\mathsf{out}}$ – the cipherext CT' is encrypted under the set of attributes $\mathcal{S}$. EN which received the transformation public key $\mathtt{tpk}$ first sets $\mathsf{I} = \mathsf{i} | \rho(\mathsf{i}) \in \mathcal{S}'$, and calculates the reconstruction of constants $\mu_{\mathsf{i} \in \mathsf{I}} = \mathsf{Recon}((\mathsf{A}, \rho), \mathcal{S})$. The decryption key corresponding to the $\mathtt{LSSS}$ scheme w.r.t. (A, ρ) is parsed as $\mathtt{tpk} = \{(\mathsf{D}_\mathsf{i})^{\frac{1}{z}}, ((\mathsf{K}_{\mathsf{i},\mathsf{j}})_{\mathsf{j}=0}^{\mathsf{n}}\}_{\mathsf{i}=0}^{\mathsf{l}})^{\frac{1}{z}}$. Then, EN computes the polynomial on the variable α with degree $\mathsf{m} + \mathsf{t} - 1$ as follows:

    $$P_{i,A(\alpha)} = \frac{\lambda_i}{\alpha}(\prod_{j=1, j \neq i} (\alpha + \mathcal{H}(a_j)) - \prod_{j=1, j \neq i} \mathcal{H}(a_j)))$$

    EN calculates $\mathsf{h}^{\frac{\mathsf{P}_{\mathsf{i},\mathsf{A}(\alpha)}}{\mathsf{z}}}$ according to the transformation public key component $((\mathsf{K}_{\mathsf{i},\mathsf{j}})_{\mathsf{j}=0}^{\mathsf{n}})^{\frac{1}{z}}$. Afterwards, EN computes Y_i which can be retrieved based on two cases w.r.t. the $\mathtt{ind}$ operator value, such that:

    - **Case 1:** if attributes have been added to the access policy:

      $$\begin{aligned} \mathsf{Y}_\mathsf{i} &= (\hat{\mathsf{e}}(\mathsf{C}_1, \mathsf{h}^{\frac{\mathsf{P}_{\mathsf{i},\mathsf{A}(\alpha)}}{\mathsf{z}}}) \cdot \hat{\mathsf{e}}((\mathsf{D}_\mathsf{i})^{\frac{1}{z}}, \mathsf{C}_2))^{\frac{1}{\prod_{\mathsf{j}=1,\mathsf{j}\neq\mathsf{i}} \mathcal{H}(\mathsf{a}_\mathsf{i}))}} \\ &= \hat{\mathsf{e}}(\mathsf{g}, \mathsf{h})^{\frac{\mathsf{s}\lambda_\mathsf{i}}{\mathsf{z}}} \end{aligned}$$

    - **Case 2:** if attributes have been revoked from the access policy:

      $$\begin{aligned} \mathsf{Y}_\mathsf{i} &= (\hat{\mathsf{e}}(\mathsf{C}_1', \mathsf{h}^{\frac{\mathsf{P}_{\mathsf{i},\mathsf{A}(\alpha)}}{\mathsf{z}}}) \cdot \mathsf{e}((\mathsf{D}_\mathsf{i})^{\frac{1}{z}}, \mathsf{E}_0'))^{\frac{1}{\prod_{\mathsf{j}=1,\mathsf{j}\neq\mathsf{i}} \mathcal{H}(\mathsf{a}_\mathsf{i}))}} \\ &= \hat{\mathsf{e}}(\mathsf{g}, \mathsf{h})^{\frac{\mathsf{s}\mathsf{F}(\alpha)\lambda_\mathsf{i}}{\mathsf{Z}}} \end{aligned}$$

    Finally, EN computes $\mathsf{Y} = \prod_{\mathsf{i} \in \mathsf{I}} \mathsf{Y}_\mathsf{i}^{\mu_\mathsf{i}} = \hat{\mathsf{e}}(\mathsf{g}, \mathsf{h})^{\frac{\mathsf{s}}{\mathsf{z}}}$ and returns it to the user U.

  - decrypt – This algorithm includes two steps. The first step, denoted by (i), enables U to retrieve the plaintext, while the second step (ii) permits to verify the correctness of the partially decrypted message received from EN:

    (i) First, U uses Y to retrieve the plaintext. Based on the partially decrypted ciphertext Y, U performs only one exponentiation without calculating any pairing functions to recover the message. M can be retrieved based on two cases w.r.t. the $\mathtt{ind}$ operator value, such that:

- **Case 1:** in the case of adding attributes to the access policy:

$$\begin{aligned} R &= \frac{C}{(Y)^{tsk}} \\ &= \frac{R \cdot (\hat{e}(g,h)^{s}}{(\hat{e}(g,h)^{\frac{s}{z}})^{z}} \\ &= \frac{R \cdot (\hat{e}(g,h)^{s}}{\hat{e}(g,h)^{s}} \end{aligned}$$

- **Case 2:** in the case of revoking attributes from the access policy:

$$\begin{aligned} R &= \frac{C'}{(Y)^{tsk}} \\ &= \frac{R \cdot (\hat{e}(g,h)^{sF(\alpha)}}{(\hat{e}(g,h)^{\frac{sF(\alpha)}{z}})^{z}} \\ &= \frac{R \cdot (\hat{e}(g,h)^{sF(\alpha)}}{\hat{e}(g,h)^{sF(\alpha)}} \end{aligned}$$

Note that if no changes have been made to the access policy and the corresponding ciphertext, the decryption process follows **Case 1**.

(ii) To retrieve the message M, U first computes $R_0 = \mathcal{H}(R)$. Then, he computes $\mathcal{H}_2(R_0||C_K)$ and compares it against V. If $V \neq \mathcal{H}_2(M_0||C_K)$, then decrypt returns $\perp$. Otherwise, he computes the symmetric key $K = \mathcal{H}_1(R)$, then decrypts the message $M = dec_K(K, C_K)$.

6 | SECURITY ANALYSIS

The security of CUPS relies on the following Theorems.

Theorem 1. ***Correctness.*** The correctness property requires that for all security parameter ξ, all attribute universe descriptions $\mathbb{U}$, all $(\mathtt{pp}, \mathtt{msk}) \in \mathtt{setup}(\xi)$, all $(\mathcal{S}, \mathcal{U}) \subseteq \mathbb{U}$ (i.e; $\mathbb{U}$ is the attribute universe), all $\mathtt{sk} \in \mathtt{keygen}(\mathtt{pp}, \mathtt{msk}, \Psi)$, all $M \in \mathbb{M}$ (i.e; $\mathbb{M}$ is the message universe), all $\Psi \in \mathcal{G}$ ($\mathcal{G}$ is the access policy space), all $CT \in \mathtt{encrypt}(\mathtt{pp}, \mathcal{S}, M)$, and all $CT' \in \mathtt{update}(\mathtt{pp}, CT, \mathtt{ind}, \mathcal{U})$, all transformation keys $\mathtt{tk} \in \mathtt{transform}(\mathtt{pp}, \mathtt{sk}, \Psi)$, all partially decyrpted messages $Y \in \mathtt{decrypt_{out}}(\mathtt{pp}, \mathtt{tpk}, CT')$, if the user has correctly obtained the secret key sk related to the Ψ required access policy $\mathcal{S}'$ for deciphering the encrypted message, the $\mathtt{derypt}(\mathtt{tsk}, Y)$ outputs M.

Theorem 2. ***Confidentiality.*** The proposed CUPS scheme is indistinguishable against replayable chosen ciphertext attacks, w.r.t. the GDHE assumption.

Theorem 3. ***Verifiability*** If $\mathcal{H}_0$ and $\mathcal{H}_2$ are collision-resistant hash functions, then the proposed CUPS scheme is verifiable against lazy and malicious edge nodes.

Here-after, we start by proving the correctness of the proposed scheme. Afterwards, we introduce the security proofs related to security games presented in Section 2.5.

6.1 | Correctness

In this subsection, we show the correctness of CUPS w.r.t. Theorem 1, while detailing the update process in subsection 6.1.1 and the decryption algorithms in subsection 6.1.2.

6.1.1 | Update Correctness

A user U who possesses a set of attributes expressed with respect to an access structure Ψ, satisfying $\mathcal{S}'$ first sets $P_{i,A(\alpha)} = \frac{\lambda_i}{\alpha}(\prod_{j=1,j\neq i}(\alpha + \mathcal{H}(a_j)) - \prod_{j=1,j\neq i}\mathcal{H}(a_j)))$. Then, U uses his secret keys to compute Y_i with respect to the two following cases:

- **Case 1:** if ind = add and $\mathcal{S}' = \mathcal{S} \cup \mathcal{U}$, then Y_i is computed as:

$$
\begin{aligned}
Y_i &= [\hat{e}(C_1', h^{P_{i,A(\alpha)}}) \cdot \hat{e}(D_i, E_0')]^{\frac{1}{\prod_{j=1,j\neq i} \mathcal{H}(a_j))}} \\
&= [\hat{e}(g^{-\alpha s}, h^{P_{i,A(\alpha)}}) \cdot \hat{e}(g^{\frac{\lambda_i}{\alpha+\mathcal{H}(\rho(i))}}, E_0^{F(\alpha)})]^{\frac{1}{\prod_{j=1,j\neq i} \mathcal{H}(a_j))}} \\
&= [\hat{e}(g^{-\alpha s}, h^{P_{i,A(\alpha)}}) \cdot \hat{e}(g^{\frac{\lambda_i}{\alpha+\mathcal{H}(\rho(i))}}, h^{s\cdot\prod_{a_j\in\mathbb{A}}(\alpha+\mathcal{H}(a_j))F(\alpha)})]^{\frac{1}{\prod_{j=1,j\neq i} \mathcal{H}(a_j))}} \\
&= [\hat{e}(g,h)^{-\alpha s P_{i,A(\alpha)}} \cdot \hat{e}(g,h)^{s\lambda_i F(\alpha)\cdot\prod_{j=1,j\neq i}(\alpha+\mathcal{H}(a_j))})]^{\frac{1}{\prod_{j=1,j\neq i} \mathcal{H}(a_j))}} \\
&= [\hat{e}(g,h)^{-s\lambda_i(\prod_{j=1,j\neq i}(\alpha+\mathcal{H}(a_j))}\hat{e}(g,h)^{s\lambda_i \prod_{j=1,j\neq i} \mathcal{H}(a_j)} \cdot \hat{e}(g,h)^{s\lambda_i F(\alpha)\cdot\prod_{j=1,j\neq i}(\alpha+\mathcal{H}(a_j))}]^{\frac{1}{\prod_{j=1,j\neq i} \mathcal{H}(a_j)}} \\
&= [\hat{e}(g,h)^{s\lambda_i \prod_{j=1,j\neq i} \mathcal{H}(a_j)}]^{\frac{1}{\prod_{j=1,j\neq i} \mathcal{H}(a_j)}} \\
&= \hat{e}(g,h)^{s\lambda_i}
\end{aligned}
$$

Afterwards, the user U computes:

$$
\begin{aligned}
Y &= \prod_{i\in I} Y_i^{\mu_i} \\
&= \prod_{i\in I} \hat{e}(g,h)^{s\lambda_i\mu_i} \\
&= \hat{e}(g,h)^{s\sum_{i\in I}\lambda_i\mu_i} \\
&= \hat{e}(g,h)^{s}
\end{aligned}
$$

Recall that the constants $mu_{i\in I}$ are the reconstruction of the LSSS matrix $\mu_{i\in I} = \mathrm{Recon}((A,\rho),\mathbb{A})$. Therefore, the user retrieves Y using the following equation

$$< \lambda, \mu >= \sum_{i\in I} \lambda_i\mu_i = \sum_{i\in I} \beta_1 = \sum_{i\in I} 1 = 1$$

- **Case 2:** if ind = revoke and $\mathcal{S}' = \mathcal{S} \setminus \mathcal{U}$, Y_i is computed as follows:

$$
\begin{aligned}
Y_i &= [\hat{e}(C_1', h^{P_{i,A(\alpha)}}) \cdot \hat{e}(D_i, E_0')]^{\frac{1}{\prod_{j=1,j\neq i} \mathcal{H}(a_j))}} \\
&= [\hat{e}(u_1^{-sF(\alpha)}, h^{P_{i,A(\alpha)}}) \cdot \hat{e}(g^{\frac{\lambda_i}{\alpha+\mathcal{H}(\rho(j))}}, h^{s\cdot\prod_{a_j\in\mathbb{A}\setminus\mathbb{U}}(\alpha+\mathcal{H}(a_j))F(\alpha)})]^{\frac{1}{\prod_{j=1,j\neq i} \mathcal{H}(a_j))}} \\
&= [\hat{e}(g,h)^{-s\lambda_i F(\alpha)(\prod_{j=1,j\neq i}(\alpha+\mathcal{H}(a_i))-\prod_{j=1,j\neq i}\mathcal{H}(a_i)))} \cdot \hat{e}(g,h)^{s\lambda_i\cdot\prod_{j=1,j\neq i}(\alpha+\mathcal{H}(a_j))F(\alpha)})]^{\frac{1}{\prod_{j=1,j\neq i} \mathcal{H}(a_j))}} \\
&= [\hat{e}(g,h)^{-s\lambda_i F(\alpha)(\prod_{j=1,j\neq i}(\alpha+\mathcal{H}(a_i))} \cdot \hat{e}(g,h)^{s\lambda_i F(\alpha)\prod_{j=1,j\neq i}\mathcal{H}(a_i)))} \cdot \hat{e}(g,h)^{s\lambda_i F(\alpha)\cdot\prod_{j=1,j\neq i}(\alpha+\mathcal{H}(a_j)))}]^{\frac{1}{\prod_{j=1,j\neq i} \mathcal{H}(a_j))}} \\
&= \hat{e}(g,h)^{s\lambda_i F(\alpha))\frac{\prod_{j=1,j\neq i}\mathcal{H}(a_i)}{\prod_{j=1,j\neq i}\mathcal{H}(a_j)}} \\
&= \hat{e}(g,h)^{sF(\alpha)\lambda_i}
\end{aligned}
$$

Afterwards, the user computes:

$$
\begin{aligned}
Y &= \prod_{i\in I} Y_i^{\mu_i} \\
&= \prod_{i\in I} \hat{e}(g,h)^{sF(\alpha)\lambda_i\mu_i} \\
&= \hat{e}(g,h)^{sF(\alpha)\sum_{i\in I}\lambda_i\mu_i} \\
&= \hat{e}(g,h)^{sF(\alpha)}
\end{aligned}
$$

Recall that the constants $mu_{i\in I}$ are the reconstruction of the LSSS matrix $\mu_{i\in I} = \mathrm{Recon}((A,\rho),\mathbb{A})$. Therefore, the user retrieves Y using the following equation

$$< \lambda, \mu >= \sum_{i\in I} \lambda_i\mu_i = \sum_{i\in I} \beta_1 = \sum_{i\in I} 1 = 1$$

6.1.2 | Decryption Correctness

We assume a data user U having an access policy Ψ which satisfies the encryption set of attributes $\mathcal{S}$. In the following, we prove the correctness of CUPS, i.e., U can retrieve the plaintext message using his access policy and the related secret keys.

First, U derives the transformation keys $\mathtt{tk} = (\mathtt{tsk}, \mathtt{tpk})$. Afterwards, he forwards the public transformation key tpk to EN.

This algorithm includes two steps. The first step, denoted by (i), enables U to retrieve the plaintext, while the second step (ii) permits to verify the correctness of the partially decrypted message received from EN:

(i) Based on the partially decrypted ciphertext Y, U performs only one exponentiation without calculating any pairing functions to recover the random message. R can be retrieved based on two cases w.r.t. the ind operator value, such that:

- **Case 1:** in the case of adding attributes to the access policy:

$$\begin{aligned} R &= \frac{C}{(Y)^{tsk}} \\ &= \frac{R \cdot (\hat{e}(g,h)^{s}}{(\hat{e}(g,h)^{\frac{s}{z}})^{z}} \\ &= \frac{R \cdot (\hat{e}(g,h)^{s}}{\hat{e}(g,h)^{s}} \end{aligned}$$

- **Case 2:** in the case of revoking attributes from the access policy:

$$\begin{aligned} R &= \frac{C'}{(Y)^{tsk}} \\ &= \frac{R \cdot (\hat{e}(g,h)^{sF(\alpha)}}{(\hat{e}(g,h)^{\frac{sF(\alpha)}{z}})^{z}} \\ &= \frac{R \cdot (\hat{e}(g,h)^{sF(\alpha)}}{\hat{e}(g,h)^{sF(\alpha)}} \end{aligned}$$

Note that if no changes have been made to the access policy and the corresponding ciphertext, the decryption process follows **Case 1**.

(ii) To verify that the retrieved message M is correct, U first computes $M_0 = \mathcal{H}(M)$. Then, he computes $\mathcal{H}_2(M_0||C_K)$ and compares it against V. If $V \neq \mathcal{H}_2(M_0||C_K)$, then $\mathsf{decrypt}$ returns $\perp$. Otherwise, EN has executed $\mathsf{decrypt}$ correctly and the retrieved plaintext is verified. Therefore, he computes the symmetric key $K = \mathcal{H}_1(R)$, then decrypts the message $M = \mathsf{dec}_K(K, C_K)$.

6.2 | Confidentiality

In the following proof, we prove that our CUPS scheme is CPA-Secure against non-adaptive Chosen Ciphertext Attacks with respect to Theorem 4.

Theorem 4. For any adversary $\mathcal{A}$, against CPA-Secure against non-adaptive chosen ciphertext, our CUPS scheme is indistinguishable according to Definition 2.5 with respect to the hardness of the General Diffie-Hellman Exponent (GDHE) assumption (Definition 4.3)

Proof. To decrypt a ciphertext CT' associated with an updated access policy $\mathcal{S}'$, $\mathcal{A}$ must recover $\hat{e}(g,h)^{s}$, in case of attributes' addition and $\hat{e}(g,h)^{s\sum_{i=0}^{t} f_i\alpha^{i}}$, in case of attributes' revocation, where the secret sharing key s is embedded in the ciphertext.

To prove that our scheme is secure against selective non-adaptive chosen ciphertext attacks, we first consider that $\mathcal{A}$ is running the Exp^{conf} experiment with an entity $\mathcal{B}$. This latter is running the $\mathsf{Exp}_{\mathcal{B}}$. Wang et al. security game [39], with $\mathcal{C}$. The objective of this proof is to show that the advantage of $\mathcal{A}$ to win the $G^{S-CPA}(1^{\kappa})$ security game is equivalent to the advantage of $\mathsf{Exp}_{\mathcal{B}}$ to win the Wang et al. security game [39]. Hereafter, $\mathcal{A}$ and $\mathcal{B}$ proceed as follows:

INITIALISATION – in this phase, the adversary $\mathcal{A}$ gives the algorithm $\mathcal{C}$ a challenge set of attributes $\mathcal{S}^*$.

SETUP – the challenger $\mathcal{C}$ runs the $\mathtt{setup}(\xi)$ algorithm, sends the public parameters $\mathtt{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, v = \hat{e}(g,h), g^{\alpha}, h, h^{\alpha}, \cdots, h^{\alpha^k}, \{u_i = g^{\alpha^i}\}_{i=1\cdots k})$ to $\mathcal{B}$ and keeps secret $\mathtt{msk}$. Consequently, $\mathcal{B}$ sends $\mathtt{pp}$ to $\mathcal{A}$.

QUERY PHASE 1 – $\mathcal{B}$ sets an empty table T and queries an access policy $\Psi_{\mathcal{A},i}$, for each session i. That is, $\mathcal{B}$ uses $\mathcal{C}$ to derive and send the queried secret keys to $\mathcal{A}$. The challenger $\mathcal{C}$ answers by running the $\mathsf{keygen}(\mathtt{pp}, \mathtt{msk}, \Psi_{\mathcal{A},i})$ algorithm. The challenger $\mathcal{C}$ generates shares of 1 relying on the

$\mathtt{LSSS}$ schema w.r.t. (A, ρ). It chooses a column vector $\beta = (\beta_1, \beta_2, \cdots, \beta_n)^T$, while $\beta_1 = s = 1$ and $\beta_2, \cdots, \beta_n \in \mathbb{Z}_p$. Then for each $i = 1$ to l, it calculates $\lambda_i =< A_i, \beta^T >$, and sets sk as $sk_{\mathcal{A},i} = \{g^{\frac{\lambda_i}{\alpha+\mathcal{H}(\rho(i))}}, (h^{\lambda_i \alpha^j})_{j=0}^{n}\}_{i=0}^{l}$.

Note that the access policy $\Psi_{\mathcal{A},i}$ does not satisfy the encryption attribute set $\mathcal{S}^*$. The private keys $sk_{\mathcal{A},i} = \{g^{\frac{\lambda_i}{\alpha+\mathcal{H}(\rho(i))}}, (h^{\lambda_i \alpha^j})_{j=0}^{n}\}_{i=0}^{l}$ are returned to $\mathcal{B}$. Subsequently, $\mathcal{B}$ sets a new entry with the private key and returns $sk_{\mathcal{A},i} = \{g^{\frac{\lambda_i}{\alpha+\mathcal{H}(\rho(i))}}, (h^{\lambda_i \alpha^j})_{j=0}^{n}\}_{i=0}^{l}$ to $\mathcal{A}$.

CHALLENGE PHASE – during the challenge phase, $\mathcal{A}$ picks two equal length cleartexts $M_0{}^*$ and $M_1{}^*$ as well as an attribute set $\mathbb{U}^*$ with $|\mathbb{U}^*| = t$ $\mathbb{U}^* \cap \mathcal{S}^* = \emptyset$ if ind = add or $\mathbb{U}^* \subset \mathcal{S}^*$ if ind = revoke. Subsequently, $\mathcal{B}$ selects $\mathcal{S}_\mathcal{B}$ such that $\mathcal{S}_\mathcal{B} \subseteq \mathcal{S}'^*$, such as $\mathcal{S}^{'*} = \mathcal{S}^* \setminus \mathcal{U}^*$ for ind = add or $\mathcal{S}^{'*} = \mathcal{S}^* \cap \mathcal{U}^*$ for ind = revoke.

Afterwards, $\mathcal{B}$ sends the access structure $\mathcal{S}_\mathcal{B}$ and the two equal length messages M_0 and M_1, defined by $\mathcal{A}$ to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ chooses a random bit b from $\{0, 1\}$ with $\mathcal{S}'_\mathcal{B} = \mathcal{S}_\mathcal{B} \setminus \mathcal{U}^*$ for ind = add or $\mathcal{S}'_\mathcal{B} = \mathcal{S}_\mathcal{B} \cap \mathcal{U}^*$ for ind = revoke and computes the challenge encrypted message $CT_b{}^* = \mathtt{encrypt}(pp, \mathcal{S}'_\mathcal{B}, M_b{}^*)$.

$$\begin{cases} E_0 = h^{s \cdot \prod_{a_i \in \mathcal{S}'_\mathcal{B}} (\alpha + \mathcal{H}(a_i))} \\ E_1 = E_0^{\alpha}, \cdots, E_{p-m} = E_{p-m-1} \\ C_1 = u_1^{-s} =, \cdots, C_{r+1} = u_{r+1}^{-s} \\ C = R \cdot v^s = R \cdot \hat{e}(g, h)^s \end{cases}$$

The challenger $\mathcal{C}$ gives $CT_b{}^*$ to the adversary if $\mathcal{U} = \emptyset$, otherwise $CT_b'^* = \mathtt{update}(pp, CT_b{}^*, \mathtt{ind}, \mathcal{U}^*)$.

QUERY PHASE 2 – in this phase, the adversary $\mathcal{A}$ can query a polynomially bounded number of queries as in QUERY PHASE 1, except that $\mathcal{A}$ cannot query secret keys related to a set of attributes $\mathcal{S}^*$.

Hereafter, two cases are considered w.r.t. the ind operator value, randomly selected by $\mathcal{C}$ in order to encrypt the challenging message such that:

- **Case 1** – the first case corresponds to attributes' addition, such that $\mathcal{C}$ sets $\mathcal{S}'_\mathcal{B} = \mathcal{S}_\mathcal{B} \cup \mathcal{U}^*$ and outputs an encrypted message CT_b', as defined in section 5.2. In this case, we first show that how a challenge ciphertext should be produced. In fact, given a ciphertext CT encrypted w.r.t. a set of attributes $\mathcal{S}_\mathcal{B}$ and $\mathcal{U}^* = \{a_1', \cdots, a_t'\}$ a new set of attributes where $\mathcal{U} \cap \mathcal{S} = \emptyset$, $\mathcal{C}$ has to add elements of $\mathcal{U}^*$ to the set of encrypting attributes $\mathcal{S}_\mathcal{B}$ of the ciphertext CT_b'. To do so, it proceeds as follows:
  Let $F(x)$ be the polynomial in x defined as $F(x) = \prod_{a_i \in \mathcal{U}^*}(x + \mathcal{H}(a_i')) = f_t x^t + f_{t-1}x^{t-1} + \cdots + f_0$ Then, the algorithm computes $E'^*{}_0 = E^*{}_0^{F(\alpha)} = \prod_{i=0}^{t} E_i^{f_i}$. The new ciphertext is then defined as $CT'_b = (E'^*{}_0, C_1, C)$ w.r.t. $\mathcal{S}'$, the new set of encrypting attributes defined as $\mathcal{S}'_\mathcal{B} = \mathcal{S}_\mathcal{B} \cup \mathcal{U}^*$.

- **Case 2** – the second case corresponds to attributes' revocation, such that $\mathcal{C}$ defines $\mathcal{S}'_\mathcal{B} = \mathcal{S}_\mathcal{B} \setminus \mathcal{U}^*$ and outputs an encrypted message CT_b', as detailed in section 5.2. That is, given a ciphertext CT encrypted w.r.t. a set of attributes $\mathcal{S}'_\mathcal{B}$ and a revocation attribute set $\mathcal{U}^* = \{a_1', \cdots, a_t'\} \subseteq \mathcal{S}'_\mathcal{B}$ where $t \leq r$, the server updates the ciphertext CT_b' as follows: Let $F(x)$ be the polynomial in x as $F(x) = \frac{1}{\prod_{a_i' \in \mathcal{U}^*} \mathcal{H}(a_i')} \prod_{a_i' \in \mathcal{U}}(x + \mathcal{H}(a_i')) = f_t x^t + f_{t-1}x^{t-1} + \cdots + f_0$. Similarly, the new ciphertext is then defined as $CT'_b = (E'^*{}_0, C'^*{}_1, C'^*{}_M)$ w.r.t. $\mathcal{S}'$, the new set of encrypting attributes defined as $\mathcal{S}'_\mathcal{B} = \mathcal{S}_\mathcal{B} \setminus \mathcal{U}^*$.

Without loss of generality, the distribution of the received challenge ciphertext does not depend on the attributes' addition and revocation. More precisely, the distribution of the challenge enciphered message is quite similar in both cases. Thus, the resistance of CUPS scheme against CPA, follows Wang et al. construction[39] construction, w.r.t. to $\mathcal{B}$, that is proven secure under the GDDHE assumption. Thus, the view of $\mathcal{B}$ is indistinguishable from the view of $\mathcal{A}$, considering a randomly selected enciphered message w.r.t. $\mathcal{S}'_\mathcal{B}$ referring to the updated access policy.

As such, we prove that our CUPS construction is secure against selective non-adaptive chosen ciphertexts attacks in the standard model, under the GDDHE assumption, with respect to Exp^{conf} security experiment. □

6.3 | Verifiability

Proof. The Exp^{verif} security games, presented in 2.5.2 captures the behaviour of lazyor a malicious edge node EN. That is, the goal of an adversary $\mathcal{A}$ is to forge a compromised partially decrypted ciphertext, that can be correctly verified by the challenger $\mathcal{C}$, by running $\mathtt{decrypt}$ algorithm.

To this end, we define an adversary $\mathcal{A}$ running the Exp^{verif} security game with an entity $\mathcal{B}$. This entity $\mathcal{B}$ is also running a collusion attack against hash function $\mathcal{H}$ with a challenger $\mathcal{C}$. The aim of this proof is to demonstrate that the advantage of the adversary $\mathcal{A}$ to succeed in the Exp^{verif} game is smaller than the advantage of the entity $\mathcal{B}$ to win the collusion game. In the following, we prove that CUPS is verifiable against lazy and malicious EN w.r.t Theorem 5.

Theorem 5. If $\mathcal{H}_2$ and $\mathcal{H}_0$ are two collision-resistant hash functions, then, CUPS is verifiable against lazy EN.

$\mathcal{B}$ executes the `setup` algorithm to generate the public parameters except the hash functions $\mathcal{H}_2$ and $\mathcal{H}_0$. Then, $\mathcal{B}$ executes both **Queries phase 1** and **Queries phase 2** to issue the secret and transformation keys.

During the challenge phase, $\mathcal{A}$ forwards a challenge message M^* to $\mathcal{B}$. $\mathcal{B}$ chooses a random message $R^* \in \mathbb{G}_T$ and encrypts R^* under the challenge access $\mathcal{S}^*$. Then, $\mathcal{B}$ defines $R_0^* = \mathcal{H}_0^*(R^*)$ and computes a symmetric key $K^* = \mathcal{H}_1^*(R^*)$. $\mathcal{B}$ execures the encryption of the message M^* using a symmetric encryption algorithm $\mathtt{enc}_K$ such as $C_{K*} = \mathtt{enc}_K(K^*, M^*)$. $\mathcal{C}$ defines the verification key $V^* = \mathcal{H}_2^*(R_0^* || C_K^*)$. $\mathcal{B}$ sends the computed ciphertext to $\mathcal{A}$ as a challenge ciphertext as well as the verification key V^*.

If $\mathcal{A}$ breaks the verifiability game, $\mathcal{B}$ will recover a message $M \notin \{M^*, \perp\}$ relying on the partially decryption algorithm $\mathtt{decrypt}_{\mathtt{out}}$.

Notice that the decryption algorithm outputs $\perp$ if $\mathcal{H}_2^*(R_0^* || C_K^*) \neq V^*$ where $R_0^* = \mathcal{H}_0^*(R^*)$ and $R^* = \mathtt{dec}_K(K, C_K)$. In the following we consider the two cases:

- **Case 1:** Since $\mathcal{B}$ knows (R_0^*, C_K^*), if $(R_0, C_K) \neq (R_0^*, C_K^*)$ is returned as a result, then $\mathcal{B}$ obtains a collision of the hash function $\mathcal{H}_2^*$.
- **Case 2:** If $(R_0, C_K) = (R_0^*, C_K^*)$, while R^* and R are not equal $(R^* \neq R)$. Then, $\mathcal{B}$ breaks the collision resistance condition of $\mathcal{H}_0^*$ as $\mathcal{H}_0^*(R) = R_0 = R_0^* = \mathcal{H}_0^*(R^*)$.

Consequently, using an *absurdum* reasoning, since the hash functions $\mathcal{H}_2$ and $\mathcal{H}_0$ are two collision resistant functions, CUPS is verifiable.

□

7 | PERFORMANCES ANALYSIS

In this section, we present the computation and the storage complexities of our proposed CUPS scheme. In our analysis, we are interested in the computations performed to execute the `encrypt`, `update` and the `decrypt` algorithms as presented in Table 2 . Furthermore, the size of the generated encrypted message, the size of the secret keys and the communications costs between the system entities are introduced in Table 3 .

TABLE 2 Features and Functionality Comparison of Attribute Based Encryption Schemes

Scheme	*Policy Update*	*Outsourcing*	*Verifiability*	*Access Policy*	*Type*	*Encryption Cost*	*Update Cost*	*User Decryption Cost*	*EN Decryption Cost*
Emura *et al.*[43]	✗	✗	✗	AND-Gates	CP-ABE	$2E_1 + E$	–	$2\tau_p + 3E_1$	–
Herranz *et al.*[36]	✗	✗	✗	Monotone	KP-ABE	$2E_1 + E$	–	$(n+1)E_1 + 5\tau_p + E$	–
Ge *et al.*[37]	✗	✗	✗	Threshold	CP-ABE	$5E_1 + E$	–	$4\tau_p + 2E_1(k-n)$	–
Wang *et al.*[39]	✗	✗	✗	Monotone	KP-ABE	$2E_1 + E$	–	$(n+1)E + E_1 + 2\tau_p$	–
Zuo *et al.*[52]	✗	✓	✗	Monotone	CP-ABE	$E + E_1(k+1) + 2\mathcal{O}(\mathcal{H})$	–	$4E + 4\mathcal{O}(\mathcal{H})$	$\tau_p(4 + 2k + n) + kE + \mathcal{O}(\mathcal{H})$
Lin *et al.*[53]	✗	✓	✓	Monotone	CP-ABE	$E_1(1+4n) + E + 2\mathcal{O}(\mathcal{H})$	–	$E + 2\mathcal{O}(\mathcal{H})$	$2nE + \tau_p(2n+1)$
Li *et al.*[41]	✗	✓	✓	AND gates	CP-ABE	$6E_1 + 2E + \mathcal{O}(\mathcal{H})$	–	$4\tau_p$	$E + 2\mathcal{O}(\mathcal{H})$
Belguith *et al.*[12]	✗	✓	✓	Monotone	CP-ABE	$5E_1 + E_T + 3\mathcal{O}(\mathcal{H})$	–	$E_T + 3\mathcal{O}(\mathcal{H})$	$3n\tau_p + E_T$
Jiang *et al.*[30]	✓	✗	✗	Threshold Threshold	CP-ABE CP-ABE	$(k-m+2)E_1 + E$ $(r+2)E_1 + E$	tE_1 $(2t+2)E_1 + \tau_p$	$nE_1 + 2\tau_p$ $nE_1 + 2\tau_p$	– –
Belguith *et al.*(The conference Version)	✓	✗	✗	Monotone	KP-ABE	$(k-m+2+r)E_1 + E$	$tE_1/(2t+2)E_1 + \tau_p$	$2\tau_p + E_1 + (n+1)E$	–
CUPS	✓	✓	✓	Monotone	KP-ABE	$(k-m+2+r)E_1 + E + 1$	$tE_1/(2t+2)E_1 + \tau_p$	$2\tau_p + E_1 + nE$	$E + 2\mathcal{O}(\mathcal{H})$

7.1 | Storage and Communication Complexities

Emura et al.[43] introduced a KP-ABE scheme requiring only 2 keys per user independent of the users' attributes. In addition, this scheme generates a ciphertext composed only of 3 elements.

Herranz et al.[36] have proposed the first CP-ABE scheme generating a ciphertext whose size does not depend on the number of attributes used in the threshold access policy. In this scheme, the user needs $k + n$ secret keys' where k is the cardinal of the attributes universes and n is the number of the users' attributes.

In [37], the authors proposes a CP-ABE scheme with constant ciphertext size. However, this schemes requires the use of $3k - 2 + n$ secret keys.

Similarly, the authors in [39] proposed a KP-ABE scheme which produces only 3 elements in the ciphertext. The users' secret keys are equal to $k(n+1)$. This size of secret keys is due to the use of `LSSS` monotone access policies which makes the scheme more expressive than the aforementioned schemes.

TABLE 3 Comparison of Storage Costs and Communication Overheads between CUPS and Closely Related ABE Scheme

Schemes	*Outsourcing*	*Key Size*	*Transformation Key size*	*Ciphertext Size*	SysInit *Communication Cost* (bits)	Storage *Communication Cost* (bits)	Retrieval From CSP *Communication Cost* (bits)	Retrieval From EN Communication Cost (bits)
Emura *et al.*[43]	✗	2	–	3	$\lvert pp\rvert + 2\Phi$	3Ω	3Ω	–
Herranz *et al.*[36]	✗	$k+n$	–	3	$\lvert pp\rvert + (k+n)\Phi$	3Ω	3Ω	–
Ge *et al.*[37]	✗	$3k-2+n$	–	4	$\lvert pp\rvert + (3k-2+n)\Phi$	4Ω	4Ω	–
Wang *et al.*[39]	✗	$k(n+1)$	–	3	$\lvert pp\rvert + k(n+1)\Phi$	3Ω	3Ω	–
Lin *et al.*[53]	✓	$2+n$	$3+n$	$4+m$	$\lvert pp\rvert + (2+n)\Phi$	$(4+m)\Omega$	$(4+m)\Omega$	$(3+n)\Phi + (4+m)\Omega$
Zuo *et al.*[52]	✓	$k+n+1$	$k+2$	$5+k$	$\lvert pp\rvert + (k+n+1)\Phi$	$(5+k)\Omega$	$(5+k)\Omega$	$(k+2)\Phi + (5+k)\Omega$
Li *et al.*[41]	✓	2	3	8	2Φ	8Ω	8Ω	$3\Phi + 8\Omega$
Belguith *et al.*[12]	✓	$2n$	$2n+3$	$4+3m$	$\lvert pp\rvert + (4n+1)\phi$	$(4+3m)\Omega$	$(4+3m)\Omega$	$(2n+3)\phi + (3+3m)\Omega$
Jiang *et al.*[30]	✗	$n+1$ $n+1$	– –	$3+k-m/3$ $r+3/3$	$\lvert pp\rvert + (n+1)\Phi$ $\lvert pp\rvert + (n+1)\Phi$	$(3+k-m)\Omega$ $(r+3)\Omega$	3Ω 3Ω	$--$
Belguith *et al.*(Conference Version)[31]	✗	$(n+1)k$	–	$3+k-m+r/3$	$\lvert pp\rvert + (n+1)k\Phi$	$(3+k-m+r)\Omega$	3Ω	$--$
CUPS	✓	$(n+1)k$	$(n+1)k+1$	$3+k-m+r/3$	$\lvert pp\rvert + (n+1)k\Phi$	$(3+k-m+r)\Omega$	4Ω	$((n+1)k)\Phi + 3\Omega$

Although, the above schemes ensure low storage and communication costs, they do not support access policy updates. Indeed, if the access rights change with the addition or the revocation of some attributes, outsourced data need to be re-encrypted.

Jiang et al.[30], have proposed a threshold CP-ABE scheme supporting access policy update. The authors proposed two different construction. The first construction ensures the addition of attributes to the access policy. This incurs the generation of a ciphertext whose size is equal to $3+k-m$ to be forwarded to the cloud server however the final user only receives 3 elements of the ciphertext no matter how many attributes are used in the access policy. The second construction provides the ability to revoke attributes from the access policy. Therefore, the generated access policy depends on the maximum number of attributes in an attribute revocation list. Like the first construction, the user only needs three elements to decrypt data. Both the proposed construction require $n+1$ secret keys for every user.

In our CUPS scheme, we apply a compact policy update technique to ensure adding and/or removing attributes from access policies in KP-ABE schemes. Therefore, the proposed construction generates a ciphertext size equal to $3+k+r-m$. Users receives a constant ciphertext size independent from the number of attributes involved in the access policy and from the applied update procedures. CUPS relies in using monotone access policies, then the users secret keys are equal to $n+k$ elements. Therefore, the proposed CUPS scheme ensures expressiveness and policy updates while introducing comparative storage with similar ABE schemes.

Regarding the communication costs, CUPS provides reasonable communication costs compared to similar ABE schemes. It requires the same communication costs between the data owner and the cloud server as other ABE schemes supporting access policy update such as Jiang et al. scheme[30] and PU-ABE scheme[31]. However, it requires an additional communication overhead between the data user and the edge node to delegate the decryption overhead similar to ABE schemes supporting decryption outsourcing[12,52,41].

Overall, CUPS introduces two interesting features, i.e., outsourced decryption and access policy update while incurring reasonable storage and communication overheads.

7.2 | Computation Complexities

The proposed schemes in[43],[36] and[39] introduce an encryption algorithm which requires two exponentiations in $\mathbb{G}_1$ and one exponentiation in $\mathbb{G}$. Ge et al.'s scheme[37] introduces an encryption algorithm requiring 5 exponentiations in $\mathbb{G}_1$ and one exponentiation in $\mathbb{G}$.
In Emura et al.' scheme[43], the user needs to perform two pairing operations and 3 exponentiations in $\mathbb{G}_1$. Herranz et al.[36] decryption algorithm requires $n+1$ exponentiations in $\mathbb{G}_1$, 5 pairing functions and one exponentiation in $\mathbb{G}$. In[37] scheme, the users executes 4 pairing operations and $2(k-n)$ exponentiations in $\mathbb{G}_1$. Wang et al. proposed a KP-ABE scheme[39] where the decryption algorithm performs $n+1$ exponentiations in $\mathbb{G}$, one exponentiation in $\mathbb{G}_1$ and two pairing operations.

The aforementioned schemes do not ensure access policies update. Jiang et al.'s scheme is the first CP-ABE scheme supporting policy updates. In this scheme, the encryption algorithm related to the attributes addition construction require $k-m+2$ exponentiations in $\mathbb{G}_1$ and only one exponentiation in $\mathbb{G}$. In addition, the attribute revocation encryption algorithms requires $r+2$ exponentiations in $\mathbb{G}_1$ and only one exponentiation in $\mathbb{G}$. This proposal consists in executing an update algorithm by the cloud server to update the used access policy used in the encryption. Therefore, it requires t exponentiations in $\mathbb{G}_1$ to add attributes and $2t+2$ exponentiations in $\mathbb{G}_1$ and one pairing operation to revoke attributes. In[30], the decryption algorithm incurs $2\tau_p + nE_1$ overhead.

CUPS scheme requires the execution of $(k-m+2+r)$ exponentiations in $\mathbb{G}_1$ and only one exponentiation in $\mathbb{G}$. The proposed scheme requires the execution of two update functions to add attributes or revoke attributes from the access policy. To add new attributes, it requires t exponentiations in $\mathbb{G}_1$ while revoking t attributes needs to an overhead equal to $(2t+2)E_1 + \tau_p$, where t is the number of attributes to be added or removed. Our CUPS scheme requires $2\tau_p + E_1 + (n+1)E$ as a decryption overhead due to the use of monotone access policies.

Above all, our proposed CUPS scheme presents quite similar processing costs compared to those incurred by most of the reviewed ABE schemes, while providing more practical features mainly related to expressiveness, decryption delegation and policy update.

8 | CONCLUSIONS

Attribute based encryption is often used to ensure encrypted access control to outsourced data for multi-user settings. That is, in several applications, users are removed and/or added, thus, it require an efficient update of users' access rights.

In this paper, we propose CUPS, a novel opportunistic computing protocol based on a key policy attribute based encryption scheme that generates short size ciphertexts and supports access policy update and decryption outsourcing features. This protocol allows addition and/or revocation of attributes without relying on a proxy server which makes it suitable for dynamic scenario such as opportunistic networks. To suit resources constrained devices, the users are able to offload the decryption overhead to the nearest edge node to partially decrypt the ciphertext while being capable of verifying that the edge node is honest in computing the partial decryption process.

As future work, we aim to test the performances of CUPS in a real-world environment by implementing the different algorithms in an opportunistic cloud of things environment.

References

1. Sarkar Subir Kumar, Basavaraju Tiptur Gangaraju, Puttamadappa C. *Ad hoc mobile wireless networks: principles, protocols, and applications.* CRC Press; 2016.

2. Wang Xiaojie, Ning Zhaolong, Zhou MengChu, et al. A privacy-preserving message forwarding framework for opportunistic cloud of things. *IEEE Internet of Things Journal.* 2018;5(6):5281–5295.

3. Wu Yue, Zhao Yimeng, Riguidel Michel, Wang Guanghao, Yi Ping. Security and trust management in opportunistic networks: a survey. *Security and Communication Networks.* 2015;8(9):1812–1827.

4. Trifunovic Sacha, Kouyoumdjieva Sylvia T, Distl Bernhard, Pajevic Ljubica, Karlsson Gunnar, Plattner Bernhard. A decade of research in opportunistic networks: challenges, relevance, and future directions. *IEEE Communications Magazine.* 2017;55(1):168–173.

5. Wu Yu, Barnard Matthew, Ying Lei. Architecture and implementation of an information-centric device-to-device network. In: :763–771IEEE; 2015.

6. Fall Kevin. A delay-tolerant network architecture for challenged internets. *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications.* 2003;:27–34.

7. Abbas Nadeem, Asim Muhammad, Tariq Noshina, Baker Thar, Abbas Sohail. A Mechanism for Securing IoT-enabled Applications at the Fog Layer. *Journal of Sensor and Actuator Networks.* 2019;8(1):16.

8. Saleem Jibran, Hammoudeh Mohammad, Raza Umar, Adebisi Bamidele, Ande Ruth. IoT standardisation: challenges, perspectives and solution. In: :1ACM; 2018.

9. Farhan Laith, Alissa Ali E, Shukur Sinan T, Hammoudeh Mohammad, Kharel Rupak. An energy efficient long hop (LH) first scheduling algorithm for scalable Internet of Things (IoT) networks. *Sensing Technology (ICST), 2017 Eleventh International Conference on.* 2017;:1–6.

10. Belguith Sana, Kaaniche Nesrine, Mohamed Mohamed, Russello Giovanni. Coop-DAAB: Cooperative Attribute Based Data Aggregation for Internet of Things Applications. *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems".* 2018;:498–515.

11. Belguith Sana, Kaaniche Nesrine, Mohamed Mohamed, Russello Giovanni. C-ABSC: cooperative attribute based signcryption scheme for internet of things applications. *2018 IEEE International Conference on Services Computing (SCC).* 2018;:245–248.

12. Belguith Sana, Kaaniche Nesrine, Laurent Maryline, Jemai Abderrazek, Attia Rabah. PHOABE: Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted IoT. *Computer Networks.* 2018;133:141–156.

13. Bacis Enrico, Vimercati Sabrina, Foresti Sara, Paraboschi Stefano, Rosa Marco, Samarati Pierangela. Mix&Slice: Efficient access revocation in the cloud. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security.* 2016;:217–228.

14. Bacis Enrico, Vimercati Sabrina De Capitani, Foresti Sara, Paraboschi Stefano, Rosa Marco, Samarati Pierangela. Access control management for secure cloud storage. *International Conference on Security and Privacy in Communication Systems.* 2016;:353–372.

15. Kaaniche Nesrine, Laurent Maryline. Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms. *Computer Communications.* 2017;111:120–141.

16. Belguith Sana, Jemai Abderrazek, Attia Rabah. Enhancing Data Security in Cloud Computing Using a Lightweight Cryptographic Algorithm. *ICAS 2015 : The Eleventh International Conference on Autonomic and Autonomous Systems.* 2015;:98–103.

17. Belguith Sana, Kaaniche Nesrine, Jemai Abderrazek, Laurent Maryline, Attia Rabah. PAbAC: a Privacy preserving Attribute based framework for fine grained Access Control in clouds. *13th IEEE International Conference on Security and Cryptography(Secrypt).* 2016;.

18. Cui Shujie, Belguith Sana, Zhang Ming, Asghar Muhammad Rizwan, Russello Giovanni. Preserving Access Pattern Privacy in SGX-Assisted Encrypted Search. In: :1–9IEEE; 2018.

19. Kaaniche Nesrine, Laurent Maryline. Attribute based encryption for multi-level access control policies. *SECRYPT 2017: 14th International Conference on Security and Cryptography.* 2017;6:67–78.

20. Cui Shujie, Belguith Sana, De Alwis Pramodya, Asghar Muhammad Rizwan, Russello Giovanni. Collusion Defender: Preserving Subscribers' Privacy in Publish and Subscribe Systems. *IEEE Transactions on Dependable and Secure Computing.* 2019;.

21. Cui Shujie, Belguith Sana, De Alwis Pramodya, Asghar Muhammad Rizwan, Russello Giovanni. Malicious entities are in vain: Preserving privacy in publish and subscribe systems. *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE).* 2018;:1624–1627.

22. Liang Kaitai, Au Man Ho, Liu Joseph K, et al. A secure and efficient ciphertext-policy attribute-based proxy re-encryption for cloud data sharing. *Future Generation Computer Systems.* 2015;52:95–108.

23. Ge Chunpeng, Susilo Willy, Fang Liming, Wang Jiandong, Shi Yunqing. A CCA-secure key-policy attribute-based proxy re-encryption in the adaptive corruption model for dropbox data sharing system. *Designs, Codes and Cryptography.* 2018;:1–17.

24. Ion Mihaela, Russello Giovanni, Crispo Bruno. Design and implementation of a confidentiality and access control solution for publish/subscribe systems. *Computer networks.* 2012;56(7):2014–2037.

25. Belguith Sana, Cui Shujie, Asghar Muhammad Rizwan, Russello Giovanni. Secure publish and subscribe systems with efficient revocation. In: :388–394ACM; 2018.

26. Belguith Sana, Gochhayat Sarada Prasad, Conti Mauro, Russello Giovanni. Emergency Access Control Management Via Attribute Based Encrypted QR Codes. In: :1–8IEEE; 2018.

27. Ogawa Kazuto, Tamura Sakurako, Hanaoka Goichiro. Key Management for Versatile Pay-TV Services. *International Workshop on Security and Trust Management.* 2017;:3–18.

28. Nkenyereye Lewis, Park Youngho, Rhee Kyung Hyune. A secure billing protocol over attribute-based encryption in vehicular cloud computing. *EURASIP Journal on Wireless Communications and Networking.* 2016;2016(1):196.

29. Jiang Yinhao, Susilo Willy, Mu Yi, Guo Fuchun. Ciphertext-policy attribute based encryption supporting access policy update. *International Conference on Provable Security.* 2016;:39–60.

30. Jiang Yinhao, Susilo Willy, Mu Yi, Guo Fuchun. Ciphertext-policy attribute-based encryption supporting access policy update and its extension with preserved attributes. *International Journal of Information Security.* 2017;:1–16.

31. Belguith Sana, Kaaniche Nesrine, Russello Giovanni. Lightweight Attribute-based Encryption Supporting Access Policy Update for Cloud Assisted IoT. In: :135-146; 2018.

32. Sahai Amit, Waters Brent. Fuzzy identity-based encryption. In: 2005.

33. Bethencourt John, Sahai Amit, Waters Brent. Ciphertext-policy attribute-based encryption. *IEEE Symposium on Security and Privacy..* 2007;.

34. Goyal Vipul, Pandey Omkant, Sahai Amit, Waters Brent. Attribute-based encryption for fine-grained access control of encrypted data. *The 13th ACM conference on Computer and communications security.* 2006;.

35. Yao Xuanxia, Chen Zhi, Tian Ye. A lightweight attribute-based encryption scheme for the Internet of Things. *Future Generation Computer Systems.* 2015;49:104–112.

36. Herranz Javier, Laguillaumie Fabien, Ràfols Carla. Constant size ciphertexts in threshold attribute-based encryption. *International Workshop on Public Key Cryptography.* 2010;:19–34.

37. Ge Aijun, Zhang Rui, Chen Cheng, Ma Chuangui, Zhang Zhenfeng. Threshold ciphertext policy attribute-based encryption with constant size ciphertexts. *Australasian Conference on Information Security and Privacy.* 2012;:336–349.

38. Attrapadung Nuttapong, Herranz Javier, Laguillaumie Fabien, Libert Benoît, De Panafieu Elie, Ràfols Carla. Attribute-based encryption schemes with constant-size ciphertexts. *Theoretical Computer Science.* 2012;422:15–38.

39. Wang Chang-Ji, Luo Jian-Fa. A key-policy attribute-based encryption scheme with constant size ciphertext. *Computational Intelligence and Security (CIS), 2012 Eighth International Conference on.* 2012;:447–451.

40. Belguith Sana, Kaaniche Nesrine, Laurent Maryline, Jemai Abderrazak, Attia Rabah. Constant-size threshold attribute based signcryption for cloud applications. *SECRYPT 2017: 14th International Conference on Security and Cryptography.* 2017;6:212–225.

41. Li Jiguo, Sha Fengjie, Zhang Yichen, Huang Xinyi, Shen Jian. Verifiable Outsourced Decryption of Attribute-Based Encryption with Constant Ciphertext Length. *Security and Communication Networks.* 2017;2017.

42. Kaaniche Nesrine, Belguith Sana, Russello Giovanni. EMA-LAB: Efficient Multi Authorisation Level Attribute Based Access Control. In: :187–201Springer; 2018.

43. Emura Keita, Miyaji Atsuko, Nomura Akito, Omote Kazumasa, Soshi Masakazu. A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. *International Conference on Information Security Practice and Experience.* 2009;:13–23.

44. Belguith Sana, Kaaniche Nesrine, Russello Giovanni. PU-ABE: Lightweight Attribute-Based Encryption Supporting Access Policy Update for Cloud Assisted IoT. *2018 IEEE 11th International Conference on Cloud Computing (CLOUD).* 2018;:924–927.

45. Qin Baodong, Deng Robert H, Liu Shengli, Ma Siqi. Attribute-based encryption with efficient verifiable outsourced decryption. *IEEE Transactions on Information Forensics and Security.* 2015;10(7):1384–1393.

46. Lai Junzuo, Deng Robert H, Guan Chaowen, Weng Jian. Attribute-based encryption with verifiable outsourced decryption. *IEEE Transactions on Information Forensics and Security.* 2013;8(8):1343–1354.

47. Green Matthew, Hohenberger Susan, Waters Brent, others . Outsourcing the decryption of abe ciphertexts. *USENIX Security Symposium.* 2011;(3).

48. Chung Kai-Min, Kalai Yael, Vadhan Salil. Improved delegation of computation using fully homomorphic encryption. *Annual Cryptology Conference.* 2010;:483–501.

49. Gennaro Rosario, Gentry Craig, Parno Bryan. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. *Annual Cryptology Conference.* 2010;:465–482.

50. Wang Hao, Yang Bo, Wang Yilei. Server Aided Ciphertext-Policy Attribute-Based Encryption. *Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on.* 2015;:440–444.

51. Boneh Dan, Boyen Xavier, Goh Eu-Jin. Hierarchical identity based encryption with constant size ciphertext. *Annual International Conference on the Theory and Applications of Cryptographic Techniques.* 2005;:440–456.

52. Zuo Cong, Shao Jun, Wei Guiyi, Xie Mande, Ji Min. CCA-secure ABE with outsourced decryption for fog computing. *Future Generation Computer Systems.* 2016;.

53. Lin Suqing, Zhang Rui, Ma Hui, Wang Mingsheng. Revisiting attribute-based encryption with verifiable outsourced decryption. *IEEE Transactions on Information Forensics and Security.* 2015;10(10):2119–2130.

AUTHOR BIOGRAPHY

Sana Belguith is a Lecturer at School of Computing, Science and Engineering, University of Salford, Manchester, UK. Previously, she used to be a Post-Doctoral Researcher in the Department of Computer Science at The University of Auckland, New Zealand. She received her engineering degree in Computer Science from the National Engineering School of Tunisia, in 2012 and her Ph.D. degree from the Tunisia Polytechnic School, Tunisia in 2017. As part of her Ph.D. programme, she was a Visiting Fellow at Télécom SudParis, France. Her major research interests include applied cryptography, distributed systems security, privacy enhancing techniques, access control, attribute-based encryption, and searchable encryption.

Nesrine Kaaniche is a Lecturer in Cybersecurity at the Department of Computer Science, University of Sheffield, co-affiliated with the Security of Advanced Systems Research Group. Previously, she was a research member of the chair Values and Policies of Personal Information, at Telecom SudParis, Institut Polytechnique de Paris, France and an International Fellow (Aug- Nov 2016) at SRI International, San Francisco, CA, USA. She received a PhD degree on cloud storage security jointly from Sorbonne University and Telecom SudParis, France, in 2014. Her major research interests include privacy enhancing technologies and applied cryptography for distributed systems and decentralised architectures, i.e., IoT, fog, cloud, and blockchains. She served as Technical Program Committee member for several conferences, and as referee for several outstanding international journals.

Giovanni Russello is an Associate Professor in the Department of Computer Science at the University of Auckland, New Zealand. He received his M.Sc.(summa cum laude) degree in Computer Science from the University of Catania, Italy in 2000, and his Ph.D. degree from the Eindhoven University of Technology (TU/e) in 2006. After obtaining his Ph.D. degree, he moved to the Policy Group in the Department of Computing at Imperial College London, UK. His research interests include policy based security systems, privacy and confidentiality in cloud computing, smartphone security, and applied cryptography. He has published more than 60 research articles in these research areas and has two granted US Patents in smartphone security. He is an IEEE member.

□