IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 2, 2021, accepted July 21, 2021, date of publication July 28, 2021, date of current version August 6, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3101048

A Higher-Level Security Scheme for Key Access on Cloud Computing

BARIS CELIKTAS[1], IBRAHIM CELIKBILEK[2], AND ENVER OZDEMIR[1], (Member, IEEE)
[1]Cyber Security Engineering and Cryptography Department, Institute of Informatics, Istanbul Technical University, 34469 Istanbul, Turkey
[2]Information and Communications Engineering Department, Institute of Informatics, Istanbul Technical University, 34469 Istanbul, Turkey

Corresponding author: Baris Celiktas (celiktas16@itu.edu.tr)

ABSTRACT In this work, we construct a key access management scheme that seamlessly transitions any hierarchical-like access policy to the digital medium. The proposed scheme allows any public cloud system to be used as a private cloud. We consider the data owner an entity consisting of several organization units. We provide a secure method for each user of this entity to access the public cloud from both inside and outside the company's network. The idea of our key access control scheme, which is based on Shamir's secret sharing algorithm and polynomial interpolation method, is suitable especially for hierarchical organizational structures. It offers a secure, flexible, and hierarchical key access mechanism for organizations utilizing mission-critical data. It also minimizes concerns about moving mission-critical data to the public cloud and ensures that only users with sufficient approvals from the same or higher privileged users can access the key by making use of the topological ordering of a directed graph, including self-loop. Main overheads such as public and private storage needs are reduced to a tolerable level, and the key derivation is computationally efficient. From a security perspective, our scheme is both resistant to collaboration attacks and provides key indistinguishability security. Since the key does not need to be held anywhere, the problem of a data breach based on key disclosure risk is also eliminated.

INDEX TERMS Cloud security, hierarchical, interpolation, key access, key assignment, secret sharing.

I. INTRODUCTION

Digitizing several services increase demands on storage systems, large-scale computations, and hosting. In addition, advances in networking technology and administrative difficulties lead companies to outsource these services. A relatively new method called cloud computing enables users to access services from any location at any time [1]. In this work, we design a novel scheme to access a cloud storage system that runs on third parties' cloud infrastructure. The proposed method provides a secure scheme so that organizations requiring a higher level of security can use any public cloud infrastructure.

Cloud computing also includes various service models [2] such as infrastructure as a service (IaaS), where a customer consumes a provider's computing, storage, and network resources; platform as a service (PaaS) where a customer uses the provider's ready-made environments to develop, run, and manage specific applications; and software as a service (SaaS) where a customer runs software on the infrastructure of the providers. The work [3] adds a service model, which is called network as a service (NaaS), where the customers are provided transport connectivity and related network services. In addition, communication as a service (CaaS), compute as a service (CompaaS), data storage as a service (DSaaS) are defined in [4], and in this work, we focus on the DSaaS model.

Cloud deployment models are categorized as private, public, community, and hybrid cloud [2]–[4]. The public cloud is defined to be a multi-tenant environment where the cloud computing environment is shared with several other users. The private cloud is a single-tenant environment where the hardware, storage, and network are dedicated to a single user. The community cloud is provided for private use by a specific consumer community and is owned, managed, and operated by the organizations in the community. In addition, the hybrid cloud is a composition of two or more distinct cloud deployment models.

In a public cloud, generally, compliance, security, and privacy requirements can create an issue since the infrastructure

The associate editor coordinating the review of this manuscript and approving it for publication was Asad Waqar Malik.

is managed and owned by a cloud storage provider that is located off-premise. The system can be accessed by any user who pays for the service. On the other hand, in the private cloud, these requirements do not generally create an issue since the infrastructure which is managed and owned by the customer is located on-premise.

Many organizations are slowing down their overall public cloud adoption plans even though public cloud infrastructure ensures many advantages, especially in total cost [5]. In addition, they avoid public cloud due to concerns about reliability, availability, data integrity, and regulatory compliance [6], [7]. According to [8], the adoption obstacles for public cloud are availability, business continuity, data lock-in, data confidentiality, and auditability. The proposed scheme offers additional security layers to alleviate or minimize these concerns regarding transferring mission-critical data to a public cloud.

The key features of our scheme designed for data owners desiring to consume DSaaS from the public cloud are extracted from the mathematical tool of Newton's interpolation. The proposed key access control scheme will be described for an organizational unit (OU) within a company which is basically one of the several organized groups that aim to accomplish a specific function in an organization. In other words, an organizational unit is one of the several vital business functions within an institution, and the key access control scheme works similarly for the others.

There are various methods to design the organizational structure of an institution. However, it is common for all that all users within an institution do not have the same privileges. In other words, the users are grouped, and attributes are defined for each group so that users' accessibility to data is well maintained.

The privileges and rights the users have and the unit or group they belong to are the most basic elements that determine whether the secret key K can be extracted or not.

In this work, for simplicity, only one organization unit OU_1 and groups G_i under this unit are considered. The group of the highest privileged users in this unit is denoted by G_1, and the group of the second-highest privileged users is denoted by G_2, and so on. The number of groups, G_i, is determined by the data owner according to the security policy.

Due to the dynamic structure of an organization, the security policy of the company should be as flexible as possible. A user who is a member of a group G_i can also be a member of another group. The security policy and accessibility rights for users outside the network might not be the same for the users within the institution's network. The group structure for the organizational unit OU_1 is similar to hierarchical, and the proposed key access control scheme is adapted accordingly. The proposed scheme provides a structure in which a user in the higher-level group has full rights to access the data to users in the lower-level groups when the pre-determined conditions are met.

Figure 1 depicts an example of pre-determined conditions in the organizational structure defined by the data owner.

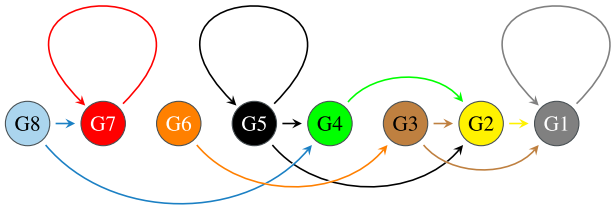

FIGURE 1. A topological ordering of a directed graph including self-loop based on the security policy defined by the data owner.

Each color represents distinct security clearance levels within the same OU. As mentioned above, G_1 has the highest level, and G_8 has the lowest level in the hierarchy. The direction of arrows indicates higher-level groups that lower-level group members need to get the approvals from to derive the K of OU.

Three distinct approval options to derive the K can be designed by the data owner. Therefore, the users can access data if they get enough approvals using one of the following options:

1) Approvals obtained from users of its own group,
2) Approvals obtained from both users of its own and higher security clearance groups,
3) Approvals should be obtained only from users of higher security clearance groups.

As seen in Figure 1, G_8 needs approvals from users of both G_7 and G_4. G_7 only needs approvals from users in G_7. G_6 only needs enough from users in G_3. G_5 needs approvals from the users in G_5, G_4, and G_2. G_4 only needs enough from users in G_2. G_3 needs approvals both from users of G_2 and G_1. G_2 only needs enough approvals from users in G_1. G_1 only needs approvals from users in G_1. In this way, the data owner can flexibly determine distinct relations according to its own security policy.

In this work, we use $G(V, E)$ for a topological ordering of a directed graph where V is a finite set of classes and a partition of set V is a collection of sets $\{V_1, V_2, \ldots, V_n\}$. E is the set of edges, and any edge can connect a node back to itself, creating a self-loop. Figure 1 is a topological ordering of a directed graph $G(8, 12)$ with a total of eight classes and twelve directed and self-loop edges. $\leq$ is a partial order (reflexive, transitive, and antisymmetric binary relation) on V, so $(V, \leq)$ is a partially ordered set (poset). Note that any two security classes V_i or groups G_i are disjoint. Let $G_i, G_j \in V$ denote two distinct classes such that $i \leq j$ for G_i and G_j. This means that users in G_i have the authority to approve data access requests (key derivation) from users G_j, consistent with the relationship in Figure 1. Note that the security clearance level of G_i is equal to or higher than that of G_j.

The remaining of the paper is structured as follows. Section 2 is devoted to the related works on hierarchical key access control schemes in the literature. Section 3 presents the architecture of the proposed scheme and explains all components in detail. Section 4 gives the implementation

results, performance and security analysis of the scheme, and other schemes. Section 5 gives a summary of the work and presents a concluding remark.

II. RELATED WORK

In this work, we develop a secure key access control scheme in the hierarchical structure to demonstrate how we can securely consume the DSaaS service. The DSaaS permits users and organizations to store an enormous amount of data on demand in a cost-effective manner [4] and [9]. With an increasing number of enterprises sharing their sensitive data on cloud servers, building a secure cloud environment for data sharing has attracted industry and academic communities.

To eliminate some of the concerns in the transition to public cloud service, key management should be done in-house, completely independent of a cloud service provider (CSP). The secure key access control should be implemented effectively and securely according to the data owner's predetermined structures. In this section, studies on hierarchical key access control schemes to provide securely accessing the data are briefly presented.

Hierarchical access control schemes [10]–[18] are based on a partially-ordered set (or poset) hierarchy. But these schemes do not provide key updates efficiently and are not practical for many groups. In addition, they do not consider the fact that the user may have access to the private key of a class for a certain period of time. From the security perspective [19], these schemes named Akl and Taylor-based can be secure under the RSA assumption.

The time-bound feature based on the Lucas function is added in the work [20] to provide better performance and time efficiency and to address key update problems. The first time-bound access control scheme is based on tamper-proofed devices is presented in [21] and the second one is based on public values [20], [22]. The first one is collusion resistant but costly and unsuitable for the cloud, limiting user convenience. But, the second one can be used efficiently in the cloud due to the characteristic of broad network access.

The other proposed schemes' main purpose is to provide secure and efficient hierarchical key access control. Thus, some parameters which measure the security and efficiency of the schemes are defined as follows:

1) The amount of private information assigned to each class to derive the secret key.
2) The amount of public information needed by the classes to derive the secret key.
3) Complexity of key derivation.
4) Complexity of key updates due to dynamic changes in the organization structure.
5) Resistance to collusion attacks known as collaborative attacks or key recovery *KR* attacks: The secret key of each class has to be protected against any coalition of users belonging to the lower classification levels; KR_s.
6) The state of key indistinguishability secure KI_s: The attacker should not distinguish between the secret key and a random string of the same length. According to [22], KI_s covers KR_s, but not vice versa.

In 1979, Shamir proposed a (k, n) secret sharing threshold scheme which enables the construction of robust key management that can function securely and reliably [23] by using a (k, n) threshold scheme with $n = 2k - 1$. The secret key K is divided into n shares. If $k - 1$ or fewer shares are available, the key cannot be constructed, and no information about K can be extracted. The idea of our proposed scheme is based on Shamir's secret sharing and is suitable for hierarchical organizational structures.

In 1983, Akl and Taylor proposed a hierarchical key assignment (HKA) scheme to manage the key and to address the hierarchical multi-group management and data sharing problem [10]. According to the proposed scheme, the users are divided into several disjoint sets $U_1, U_2, U_3 \ldots U_n$. Security classes are based on a relationship of a poset hierarchy. The users of U_k can access data held by the users in the U_n where $n \geq k$ while the other way around is impossible. The scheme can be useful in a secure distributed system. However, it does not completely solve more general multi-level security issues and cannot be adapted flexibly and dynamically. According to [22], the scheme expensively performs key derivation and only provides security regarding the *KR*. According to work, [11], due to the large number of keys held by each user, the scheme becomes inefficient as the number of users increases. In addition to this, a large amount of public storage for each security class is needed [13], [18].

In 1985 [11], an improved version of the scheme described in [10], called a canonical assignment, is proposed to reduce the amount of storage need for public information, especially when the number of classes in the hierarchy is large [20]. However, according to the work [18], the need for a large amount of storage is not eliminated. Any member of a group can access the data of the subgroup users, as they can generate the subgroup users' keys using their own keys. It also provides resistance to collusion attacks.

In 1988 [12], Sandhu proposed a tree hierarchical scheme in which security classes are organized as rooted-tree; this is a special instance of poset hierarchy. Using the iterative method of one-way function that is easy to compute, the key belonging to the lower security class in the sub-tree can be generated, but the inverse is difficult to compute. New security classes can be added without changing keys for existing classes, unlike [10] and [11]. In addition, there is also no need for extra public parameters for the key derivation. However, the main disadvantage of this algorithm [12] is the computational overhead in key derivation, especially when the root of the tree must generate the key of the lowest class. The scheme is only implemented in a tree hierarchy and is impractical for trees larger than ten security levels.

In 1990, Harn and Lin proposed a similar approach to the method in the work [10] but followed a policy of bottom-top key derivation [13]. Unlike [10], [11], new security classes can be added without changing all keys. The storage need for public parameters for security classes is much smaller

than [10], [11]. Also, the scheme is more efficient in memory usage compared to [10].

In 1993, Chang *et al.* [14] and Liaw *et al.* [15] proposed schemes based on Newton's implementation and one-way function. However, the computational time required for key generation and derivation is excessive and time-consuming. In addition, their schemes are not resistant to collusion attacks [16].

In 1997, Hwang modified the algorithm presented in [17] to be used in poset hierarchy. A higher-level user can obtain the keys of other users at lower levels from his own cryptographic key [18]. This is a one-way function, and its preimage resistant. The scheme is resistant to collusion attacks.

In 2002, Tzeng [20] proposed a time-bound cryptographic key access control scheme to prevent the key from being used by members of higher level class C_i continuously. It can be said that the scheme is the time period-based version of [10]. Any user of a class C_i can only be a member of C_i for a certain period of time and compute from secret K_i to K_j at that time t if and only if $C_j \leq C_i$ and $t_1 \leq t \leq t_2$. Note that t_1 is the beginning, t_2 is the end of time period. There are broadcasting data to authorize users in a hierarchical structure with optimal bandwidth, and unauthorized users cannot obtain any data by listening to the broadcasting. In addition, a user can keep encrypted data for a period of time, and a higher classified user can grant another user a privilege to disclose the encrypted data, which ensures flexibility. The scheme is independent of the number of classes in the hierarchy, unlike all previously proposed key access control schemes in the poset hierarchy. However, the scheme is not efficient since the users must always keep the keys in their hands to access the authorized data for a certain period of time. The scheme is storage efficient but computationally inefficient due to the need for a large number of public-key computations, overhead and implementation costs [21].

In 2004, Chien [21] proposed an efficient time-bound hierarchical key assignment scheme inspired by [10], [20]. The scheme proposes to improve the time-slot-based key assignment scheme [20] by assigning distinct cryptographic keys to solve both implementation cost and performance issues in the hierarchy. The scheme is based on a tamper-proof device that only performs simple arithmetic operations. It is computationally secure to derive the secret key from the public value. A user cannot derive any key except the authorized time slots. Users at lower clearance levels cannot obtain the key of higher clearance levels, so the scheme is collusion resistant. The scheme is more efficient and needs a low-cost tamper-proof device that supports small storage and simple operations without public-key cryptography, and has little computational complexity. However, Tzeng [20] the scheme has been proven to be insecure against a feasible and efficient collision attack if three users conspire to gain access to the keys [24], [25]. In addition, both schemes of Chien's [21] and Tzeng's [20] have been proven not to be resistant to collusion attacks if three users conspire to gain access to the keys [25].

In 2009, the scheme of Atallah *et al.* [27] based on the use of pseudo-random functions works with random access graphs; only hash functions are used to derive the descendant's key from its own key, the number of linear bit processes limits the key derivation, and the class consists of a single key associated with that class. Besides the work [27], the scheme described in [28] is also adapted for removing and insertion of classes in the hierarchy. KI_s and KR_s notions first showed up with [27]. The scheme is secure against chosen-plaintext attacks and security based on pseudo-random functions and an additional symmetric encryption scheme. The complexity of key derivation can be implemented using $O(n)$ hash functions. According to [22], each user has to store a maximum of three private data, and the amount of public data adversely affects the complexity of key derivation. According to [29], the number of public information increases with the number of edges on the graph and number of classes. As the number of levels between classes increases, the cost of key derivation increases linearly.

Elliptic-curve cryptography-based hierarchic key assignment schemes are also proposed [30]–[32]. In [30], the number of access control policies depends on the number of encryption keys and tamper-proof devices, and the resulting scheme is slower than [20], [21]. Sun *et al.* [33] has proven that the scheme [30] is not collusion resistant. In addition, Das *et al.* [34] has proven the scheme [31] is vulnerable to exterior root-finding attack. On the other hand, Lin and Hsu [35] have proven the scheme [32] is not collusion resistant.

In 2011, Santis *et al.* [36] and in 2012, Hassen *et al.* [37] proposed their schemes with better performance than the previous ones. The schemes [36] (TBEBF:time-bound encryption-based family and TBBEBF:time-bound broadcast encryption-based family) support dynamic updates in the hierarchy at the expense of an increase in the amount of public information. The schemes achieve KI_s and significantly reduce the method's [27] computational requirements for key derivation and key updates. The scheme's private key storage need is tolerable, but the amount of public storage need grows linearly as the number of classes and edges in the graph increases. On the other hand, the scheme of [37] has efficient storage, bandwidth, and computation overheads in comparison with previous ones.

In 2012, Ateniese *et al.* [22] designed two different time-bound schemes, the TLEBC (two-level encryption-based construction) is based on symmetric encryption schemes, and the TLPBC (two-level pairing-based construction) is based on bilinear maps. These schemes are provable-secure (KI_s and KR_s) and can compute the keys of all lower classes in the hierarchy more efficiently. Also, without needing to change any private information, only local changes to the public information are sufficient for updating the hierarchical structure. According to [29], private information can be as large as the number of periods because the schemes are based not only on the number of classes but also on time periods.

In 2013, the schemes based on pseudo-random functions *PRF* and forward secure pseudorandom generators *FSPRG* for arbitrary posets were designed by Freire *et al.* [29]. The ultimate efficiency of key derivation depends on the longest poset depth, whereas private information depends on the poset width. The schemes do not need public storage. Updated KI_s notion of [27] provides stronger security SKI_s than all predecessors schemes.

In 2016, a hierarchical key access (HKA) scheme based on linear geometry was proposed [38]. To derive the key of the clearance level, the public vector of its own and the private vector of the ancestor clearance level are used together. Without the need for iterative computation, the key of the descendant can be directly derived by the ancestor. The scheme that ensures SKI_s only needs to compute the vector multiplication and the values of the pseudorandom function, resulting in very little computational overhead. Although the size of the public information is slightly larger than the others, there is a trade-off between computation cost and storage. The scheme provides an efficient key management solution that can serve as flexible and fine-grained hierarchical access control to address potential changes in the hierarchy with light computations in a finite field. However, the ultimate overhead of every class is not tolerable and efficient. If there is a change in the hierarchy, the data owner must compute and publish a new public matrix. The matrix must be square to establish the relationship between the number of classes and the public information, especially for rekeying process.

In the literature, there are two types of hierarchical key access control schemes, one is indirect access schemes, and the other is direct access schemes. In indirect access control schemes such as [29], [36], [37], the secret key of the parent class can be derived by calculating all keys on the path to the child class. In direct access control schemes such as [28], [31], [35], [38] requires only one calculation to derive the secret key of the child classes. But their disadvantages are their security [34], [41], [42] and high overhead [35].

While several key access control schemes can be tailored to meet security requirements, some open areas need to be addressed, such as creating a cost-effective and scalable scheme. Therefore, the above-mentioned disadvantages regarding efficiency and security motivated us to construct a secure and effective hierarchical key access control scheme. Following the organization's security level policy, securing access to the data uploaded to the public cloud both from inside and outside the company, guaranteeing hierarchical access control mechanism on the basis of organizational unit, not sharing secret keys with any user but only through a secure channel, and the lack of the need to hold the key anywhere are the main features of the proposed scheme. Note that the data owner can change the security level policy at any time. Overall, the proposed scheme offers a dynamically adjustable, secure, hierarchical, efficient, and flexible mechanism.

III. THE ARCHITECTURE OF THE PROPOSED SCHEME

In this section, the details of the proposed scheme will be explained.

In most cases, the data owner is an organization deploying the standalone workstation within its network, and this standalone workstation consists of six components. Security Level Policy (SLP) implements the data access and control policy of the data owner. Key Establishment Unit (KEU), a core component of the system on the data owner side, executes secret key splitting, computing operations, shares generations, and performs approval and key derivation mechanisms according to inputs from LDAP queries and SLP. Credential Generator (CG) performs secret key construction by using the key components received from KEU and sends the secret key to Data Processor (DP) or Cloud Management Client (CMC). DP or CMC encrypts data before sending and decrypts data after it is downloaded from the public cloud. The Network Control Policy (NCP) checks whether the request is received inside or outside the network. Integrity Controller (IC) checks whether the data in the public cloud has been compromised at any time.

The data owner installs a CMC application for each user outside the network. The application provides secure communication with a standalone workstation inside the network and performs encryption of data before uploading and decryption of data after downloading data. Basically, CMC is an application with the same role as DP.

Suppose that the data owner has a KDC (key distribution center) responsible for generating a secret key for each organizational unit for the Computing Component. With our proposed scheme, since the KDC does not need to hold these keys anywhere, the problem of data breaches and the key disclosure risk will be eliminated.

To better explain the components and workflow, we divide the scheme into four steps, the system preparation process, the KEU operations phase, the phase of uploading data to the public cloud, and the phase of downloading data from the public cloud. Note that in the KEU operation phase, the core of the proposed scheme is directly related to secret key derivation.

A. THE SYSTEM PREPARATION PHASE

The system preparation phase is illustrated in Figure 2.

Each organization unit has only one secret key to access the cloud and perform the encryption and decryption process. For simplicity, we assume that a data owner has only one organization unit denoted by OU_1, which consists of n user groups. Let the secret key for this organization unit be k_{OU_1}. The set of user groups of OU_1 is denoted by G_k for $k = 1, \ldots, n$. Note that G_1 has the highest security clearance level, and G_n has the lowest security clearance level in the organization. The users in each group can access data in the public cloud inside or outside the network if the security conditions are met.

G_k denotes a user group, and the key component is denoted by c_k. In addition, each user in the group has a share of k_{OU_1}

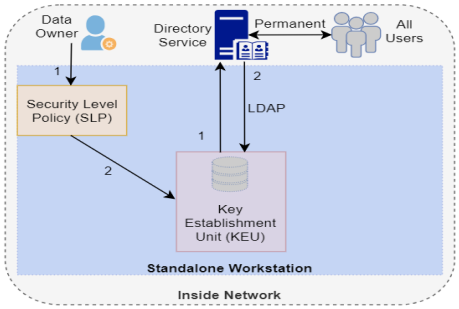

FIGURE 2. The system preparation phase.

which is distributed via (t_k, w_k)-threshold scheme defined for each c_k. Note that $|G_k|$ represents the number of users of G_k. By the data owner, P_k^I is defined for the security policy inside the network, whereas P_k^O is defined for the security policy outside the network. Each of these policies determines how group members might obtain k_{OU_1}. In other words, these policies determine how to split the k_{OU_1}. The set of key components is defined as C_k^I for inside the network, whereas C_k^O is defined for outside the network. The equation 1 and 2 defines how the C_k and k_{OU_1} can be constructed by the data owner following the P_k like Figure 1.

$$C_k = \{\delta_k * c_k, \delta_{k-1} * c_{k-1}, \ldots, \delta_1 * c_1\} \tag{1}$$

$$k_{OU_1} = \sum_{i=1}^{k} \delta_i * c_i \mod p \text{ where } \delta \in \{0, 1\} \tag{2}$$

Note that P_k defines the security clearance level for each G_k by making change in δ value.

Remember that the secret sharing for each c_k is based on (t_k, w_k) threshold scheme. Shamir's [23] (t, w) threshold scheme with $w = 2t - 1$ is utilized for this purpose and $w_k = |G_k|$. Each key component cannot be reconstructed if $t - 1$ approvals or less are received.

P_k defines (t, w) threshold scheme for each c_i in C_k as seen in the equations from 3 to 5 in accordance with the SLP defined by data owner.

The lowest-security level policy can be defined as

$$t_k = \lceil (|G_k| + 1)/2 - \beta \rceil \tag{3}$$

The intermediate-security level policy can be defined as

$$t_k = \lceil (|G_k| + 1)/2 \rceil \tag{4}$$

The highest-security level policy can be defined as

$$t_k = \lceil (|G_k| + 1)/2 + \beta \rceil \tag{5}$$

Assume that the default value of β is defined as $\lceil |G_k|/4 \rceil$, but β can easily be modified by the data owner flexibly in line with the company's constantly changing security policies.

In addition, alternate policies AP_k for each G_k can be set up by the data owner to ensure the reliability and robustness of the scheme. When the main policy P_k is failed, AP_k is replaced for each G_k. Namely, if each c_k for the P_k cannot be obtained due to the lack of approvals from the members of G_k or due to network failure or packet loss during transmission, one of the AP_k will be utilized to access the key, which guarantees fault tolerance in the hierarchy.

B. KEY ESTABLISHMENT UNIT OPERATIONS PHASE

KEU is a core of our scheme consisting of three sub-components; Computing Component (CC), Key Component Generator (KCG), and Datastore. KEU performs its operations after System Preparation Phase is completed. KEU operations are illustrated in Figure 3.

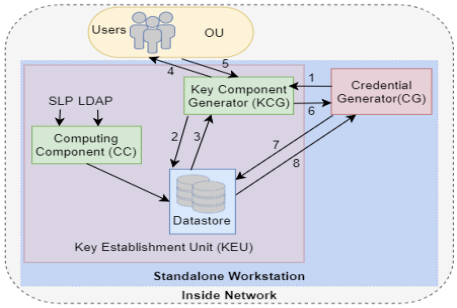

FIGURE 3. KEU operations phase.

CC determines a large size prime field $\mathbb{F}_p$ for OU_1. The mathematical operations are performed over the finite field $\mathbb{F}_p$. Each c_i in C_k is selected randomly over $\mathbb{F}_p$ and k_{OU_1} can only be generated by addition of all c_i.

CC randomly selects distinct polynomials over $\mathbb{F}_p$ for each (t_k, w_k)-threshold scheme. These schemes are defined for each c_i in C_k. Namely, a random polynomial for c_i is $p_i(x) \in \mathbb{F}_p[x]$ such that $p_i(0) = c_i$. and the degree of $p_i(x)$ is $t_i - 1$.

Each user $u_{kj} \in G_k$ has a random point/share $(x_{kj}, p_k(x_{kj}))$ on the graph of $p_k(x)$ for key component c_k. To compute c_k, it is necessary to have at least t_k number of point/share on the graph.

Users of an OU_1 can only decrypt data encrypted by a user of OU_1. However, if a user subscribes to another OU_k, the encrypted data associated with this OU_k can also be decrypted. Namely, a user can subscribe to other user groups in different units by the data owner. In this way, the user has more privileges over the data in the public cloud. This also provides a dynamic, secure structure for encrypted data in the public cloud.

CC creates a database D_k in the datastore for each OU_k. Each D_k consists of both Key Establishment Table (KET) and user tables. For each user, there are two tables for both inside and outside the network. Since we only have one organizational unit OU_1, the database is denoted by D_1.

CC computes all needed key shares for each polynomial and stores all (t_k, w_k) schemes in related user's tables in D_1.

If a new user is added to G_k, CC computes all (t_k, w_k) schemes and stores them in related user's tables. If any user is removed from a G_k, the related (t_k, w_k) schemes are deleted. If $|G_k|$ is less than t_k, System Preparation Phase will be reinitialized.

When a user wants to upload data to or download data from the public cloud, the request arrives at the CG with the parameters OU_1, G_k, and γ. When CG receives the request, it determines where the request is sent and which G_k the user is a member of. Then, CG sends a request $Req(OU_1, G_k, \gamma), \gamma \in \{I, O\}$ to KCG to obtain all c_k. If a request is received from inside the network, then $\gamma = I$ and from outside, then $\gamma = O$.

Once the KCG receives the request from CG, KCG chooses D_1 based on OU_1 and G_k, and sends a query to D_1 to learn all (t_i, w_i) for each c_i in C_k^{γ}.

Then KCG sends requests $\bigcup Req(t_i, w_i)$ to users within OU_1 in accordance with P_k in order to receive sufficient approvals. If sufficient approvals $\bigcup Rep(t_i, w_i)$ are correctly received by KCG, it calculates all c_i namely $p_i(0)$. Then KCG performs Equation 1 and submits C_k^{γ} to CG. Then CG creates k_{OU_1} using the components c_i over the $\mathbb{F}_p$ namely, it performs Equation 2, and then sends k_{OU_1} to the DP or CMC.

Note that if any other user requests a new upload to or download from the public cloud, one of the following processes starts again. By issuing tokens for a certain period of time to the requesting user or using caching, we can minimize the delay that the network can cause and optimize the time cost. However, the key derivation cost will be calculated without considering these delays and network issues in our work.

C. THE PHASE OF UPLOADING DATA TO PUBLIC CLOUD

1) INSIDE THE NETWORK

The phase of uploading data to the public cloud from inside the network is illustrated in Figure 4.

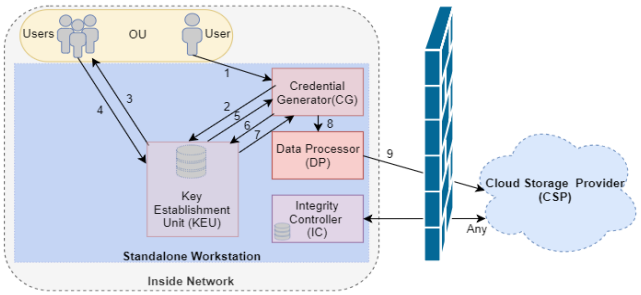

FIGURE 4. Uploading data to the public cloud, inside the network.

Suppose that a user inside the network wants to upload data to the public cloud. Then, this user will first set up a connection with CG and makes a request.

CG asks KEU to respond to the user's request. CG needs to have C_k to create k_{OU_1}. KEU requests to receive sufficient approval from other users within OU_1. If sufficient approvals are received, and security conditions are met, KEU computes C_k of k_{OU_1} and submits it to CG. CG calculates k_{OU_1} using C_k and sends k_{OU_1} to the DP. After receiving k_{OU_1}, DP encrypts the data $E_{k_{OU_1}}(data)$ and uploads the encrypted data to the public cloud.

The k_{OU_1} is only used to encrypt and decrypt the data of users in the OU_1. Then the secret key $DEL(k_{OU_1})$ is deleted automatically after a certain period of time. Note that the k_{OU_1} is not stored anywhere.

Publicly or privately verifiable methods are used as proof of storage [43]. IC uses k_{OU_1} and runs one of the privately verifiable proofs of storage protocol with the cloud storage provider to verify the integrity of the data to check any unauthorized modification.

2) OUTSIDE THE NETWORK

The phase of uploading data to the public cloud from outside the network is illustrated in Figure 5.

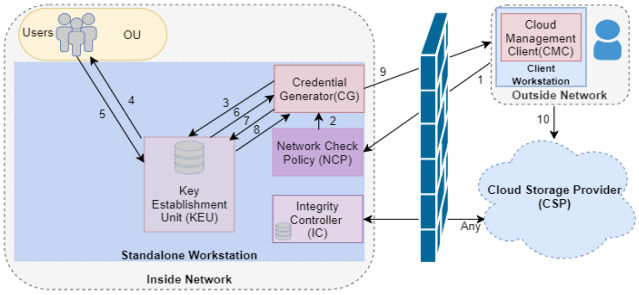

FIGURE 5. Uploading data to the public cloud, outside the network.

Suppose that a user outside the network wants to upload data to the public cloud. If the user outside of the network sends an upload request to the public cloud using CMC, then this request will first arrive at the company's NCP component of the standalone workstation. After the user authenticates itself to NCP, NCP sends the request to CG.

The same process as inside the network works again after the request received by CG.

After CG creates k_{OU_1} using C_k over the $\mathbb{F}_p$. It sends k_{OU_1} to CMC over the secure network channel. After receiving k_{OU_1}, CMC encrypts the data $E_{k_{OU_1}}(data)$ and uploads the encrypted data to the public cloud. Then CMC deletes the secret key $DEL(k_{OU_1})$ automatically after a certain period of time.

D. PHASE OF DOWNLOADING DATA FROM THE PUBLIC CLOUD

The process of downloading encrypted data from the public cloud is nearly the same as uploading encrypted data to the public cloud.

1) INSIDE THE NETWORK

Suppose that a user inside the network wants to download data from the public cloud.

When DP receives the encrypted data $E_{k_{OU_1}}(data)$ from the public cloud, it decrypts the data with k_{OU_1} received from CG. DP decrypts data by performing $D_{k_{OU_1}}(E_{k_{OU_1}}(data))$ operation. Then DP delivers the decrypted data to the user who makes a downloading request.

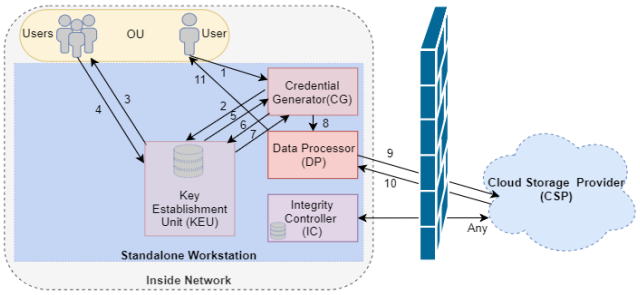

FIGURE 6. Downloading data from the public cloud, inside the network.

2) OUTSIDE THE NETWORK

Suppose that a user outside the network wants to download encrypted data from the public cloud.

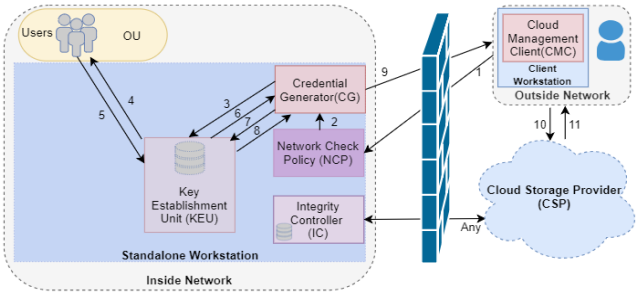

FIGURE 7. Downloading data from the public cloud, outside the network.

The request is received by NCP and CG respectively, then CG creates k_{OU_1} using C_k over the $\mathbb{F}_p$. It sends k_{OU_1} to CMC over the secure network channel. After receiving k_{OU_1}, CMC send a downloading request to Cloud Service Provider. Then CMC decrypts the encrypted data $D_{k_{OU_1}}(E_{k_{OU_1}}(data))$.

IV. IMPLEMENTATION RESULTS

In this section, we give the implementation results of our scheme and then discuss performance and security analysis, respectively. A comparison with other schemes will also be presented.

We implemented the proposed higher-level security scheme for the key access scheme on a computer with Ubuntu 20.04.2 64-bit operation system running on Intel Core i7-7567U CPU 3.50GHZ processor. It has 16 GB of memory, and we use SageMath, a free, open-source mathematics software utility.

The proposed scheme has been defined as two different algorithms. Firstly, the system will be ready by following the steps described in Algorithm 1 (system ready state and includes key establishment unit operations). Algorithm 1 calculates all threshold parameters and shares in the finite $\mathbb{F}_p$ for each user in the hierarchy based on Shamir's secret sharing algorithm. Upon constructing all parameters and if the system is ready, then Algorithm 2 or Figure 8 (for simplicity) steps will be followed to derive the K_{OU_t} for each user in the hierarchy.

Algorithm 1:

System Preparation Phase:

permanent Directory Service is running
SLP (P_k^I, P_k^O for each group) is defined by Data Owner

Key Establishment Unit Operations Phase:

choose a finite field F_p for OU_t
create a database D_t for OU_t
for $i = 1$ *to* n **do**
  create KET for G_i
  $C_i^I \leftarrow \{\}$
  $C_i^O \leftarrow \{\}$
  for $j = 0$ *to* $|P_i^I| - 1$ **do**
    $\delta_{i-j}^I = P_i^I[j]$
    if $\delta_{i-j}^I == 1$ **then**
      compute $c_{i-j} \in \mathbb{F}_p, p_{i-j}(x) \in \mathbb{F}_p[x]$
      # $p_{i-j}(0) = c_{i-j}$
      C_i^I.append(c_{i-j})
      construct (t_{i-j}, w_{i-j}) based on $p_{i-j}(x)$
  for $j = 0$ *to* $|P_i^O| - 1$ **do**
    $\delta_{i-j}^O = P_i^O[j]$
    if $\delta_{i-j}^O == 1$ **then**
      compute $c_{i-j} \in \mathbb{F}_p, p_{i-j}(x) \in \mathbb{F}_p[x]$
      # $p_{i-j}(0) = c_{i-j}$
      C_i^O.append(c_{i-j})
      construct (t_{i-j}, w_{i-j}) based on $p_{i-j}(x)$
  store all threshold schemes (t, w) and user shares.

As seen in Table 1 below, the following options to generate k_{OU_1} have been used to create a sample of implementing our proposed scheme and explained in detail below. While implementing these options, assume that each user has enough secret shares to be used to generate the k_{OU_1}. For many G_k, many options can be designed to generate the k_{OU_1} as in Table 1 by the data owner following SLP. This allows the data owner to arrange a flexible and dynamic structure. This is one of the main goals of the proposed scheme. Note that the topological ordering of a directed graph including self-loop as in Figure 1 must be defined by the data owner.

For G_1 users, in the first option, c_1 is enough to generate the k_{OU_1}, and for that, only approvals from the users of G_1 are needed, in other words, at least four approvals from seven users of G_1 are required to generate the k_{OU_1}.

For the users of G_2, in the first option, c_2 can generate k_{OU_1} and eight approvals from fifteen users are enough to generate c_2 which gives k_{OU_1}. In the second option, both c_2 and c_1 can together generate the k_{OU_1}. Therefore, at least six approvals from fifteen users in G_2 and at least one approval from seven users in G_1 are needed to generate the k_{OU_1}. In the third option, both four approvals from fifteen users in G_2 and two approvals from seven users in G_1 are needed to generate the k_{OU_1}.

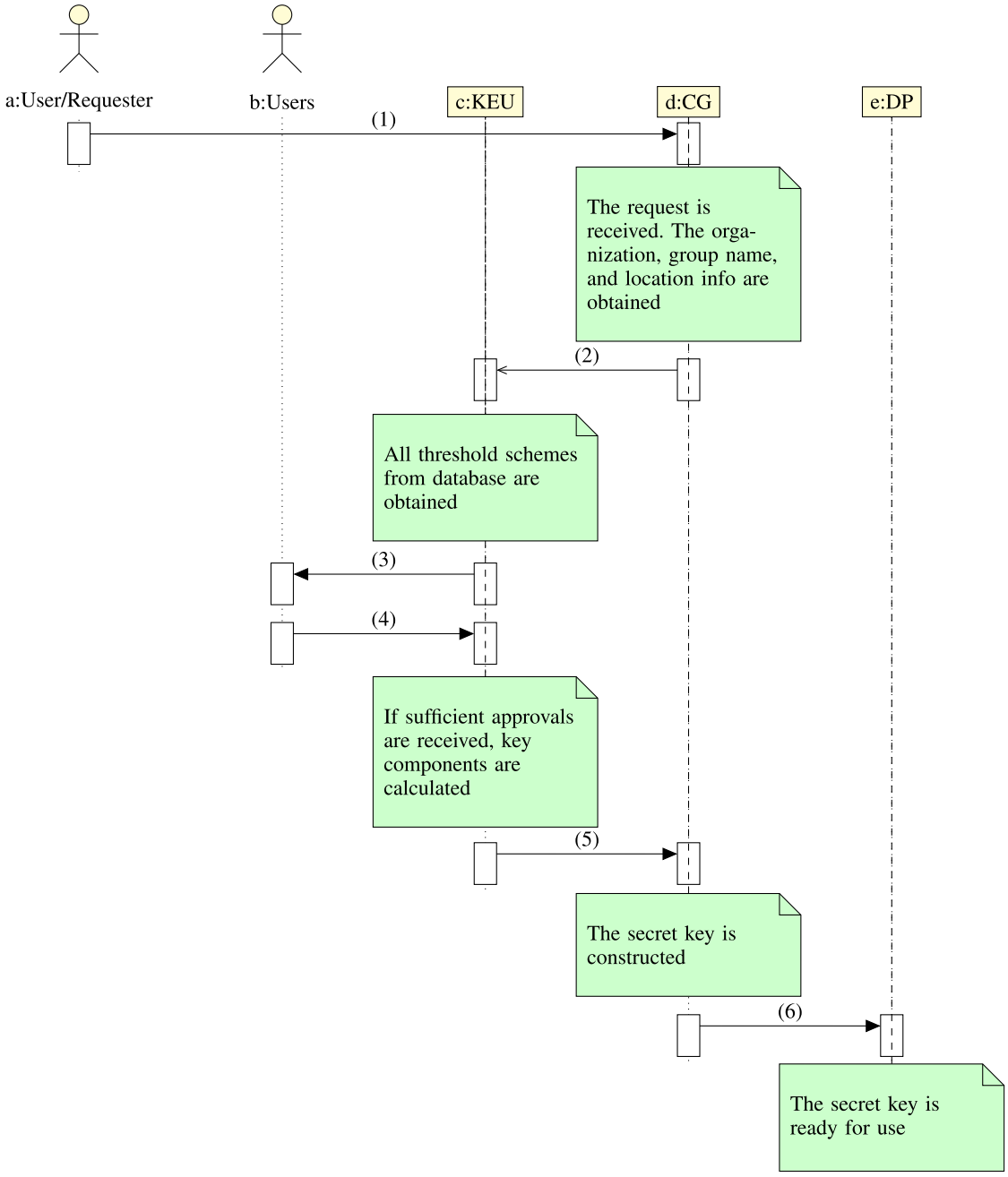

FIGURE 8. The secret key derivation phase. (1) The user/requester sends a request to access the key. (2) CG sends a request for the user/requester. (3) KEU send requests to receive sufficient approval from other users within the same organization unit. (4) Users send replies (approval or rejection). (5) KEU sends the key components. (6) CG sends the secret key.

For G_3 users, in the first option, c_3 can generate k_{OU_1} and thirteen approvals from twenty-five users are enough to generate the k_{OU_1}. In the second option, both c_3 and c_2 can together generate the k_{OU_1} and both eleven approvals from twenty-five users in G_3 and one approval from fifteen users in G_2 are needed to generate the k_{OU_1}. In the third option, both c_3 and c_1 can together generate the k_{OU_1} and both nine approvals from twenty-five users in G_3 and one approval from seven users in G_1 are needed to generate the k_{OU_1}. In the fourth option, c_3, c_2, and c_1 can together generate the k_{OU_1} and seven approvals from twenty-five users in G_3, one approval from fifteen users in G_2, and one approval from seven users in G_1 are needed to generate the k_{OU_1}.

A. PERFORMANCE ANALYSIS

The following performance analysis results are obtained considering the worst-case scenario, for example, using the k^{th} option in Table 1 (the most secure option determined by the data owner).

TABLE 1. Sample key access control scheme in organization.

User Groups	1^{st} option	2^{nd} option	3^{rd} option	4^{th} option	k^{th} option
G_1 $\|G_1\| = 7$ C_1	$\{c_1\}$ $\{(t_1, w_1)\}$ $\{(4, 7)\}$	...	...	...	...
G_2 $\|G_2\| = 15$ C_2	$\{c_2\}$ $\{(t_2, w_2)\}$ $\{(8, 15)\}$	$\{c_2, c_1\}$ $\{(t_2, w_2), (t_1, w_1)\}$ $\{(6, 15), (1, 7)\}$	$\{c_2, c_1\}$ $\{(t_2, w_2), (t_1, w_1)\}$ $\{(4, 15), (2, 7)\}$	...	...
G_3 $\|G_3\| = 25$ C_3	$\{c_3\}$ $\{(t_3, w_3)\}$ $\{(13, 25)\}$	$\{c_3, c_2\}$ $\{(t_3, w_3), (t_2, w_2)\}$ $\{(11, 25), (1, 15)\}$	$\{c_3, c_1\}$ $\{(t_3, w_3), (t_1, w_1)\}$ $\{(9, 25), (1, 7)\}$	$\{c_3, c_2, c_1\}$ $\{(t_3, w_3), (t_2, w_2), (t_1, w_1)\}$ $\{(7, 25), (1, 15), (1, 7)\}$	...
G_n $\|G_n\| = N$ C_n	$\{c_n\}$ $\{(\lceil \frac{N}{2} \rceil, N)\}$	$\{c_n, c_{n-1}\}$ $\{(\lceil \frac{N}{2} \rceil - 2, N), (1, N)\}$	$\{c_n, c_{n-2}\}$ $\{(\lceil \frac{N}{2} \rceil - 4, N), (1, N)\}$	$\{c_n, c_{n-1}, c_{n-2}\}$ $\{(\lceil \frac{N}{2} \rceil - 6, N), (1, N), (1, N)\}$	...

Algorithm 2:

Input:

1 Request $(OU_t, G_k, \gamma \in \{I, O\})$ from the user

Key Derivation Phase:

2 **for** *each request* **do**

3 $req = (OU_t, G_k, \gamma)$

4 **if** *req is legitimate* **then**

obtain all (t, w) from D_t

5 **if** *sufficient approvals are received* **then**

computes all c_i and C_k^{γ}

$K_{OU_t} \leftarrow \sum c_i \mod p$

6 return K_{OU_t}

We assume that all policy sets are constructed on the basis of $P_k = \{1\}^k$, $1 \leq k \leq n$. In other words, each component set C_k is constructed as $\{c_k, c_{k-1}, \ldots, c_1\}$.

1) PUBLIC INFORMATION STORAGE NEED

The user who is a member of a group and requesting the derivation of the key does not need to have any public information. Thus, our scheme's public information need is zero, which is one of the advantages over most other hierarchical key access control schemes in the literature.

2) PRIVATE INFORMATION STORAGE NEED

The number of groups in OU be n and G_k, $1 \leq k \leq n$ be any group in OU. Note that each key share is a random point $(x_{kj}, p_k(x_{kj}))$ on the graph of $p_k(x)$ for key component c_k and multiple shares can be stored in user tables. These shares are used to construct secret polynomials and key components and ultimately derive k_{OU}.

According to the worst-case scenario security policy, assume that users of G_1 are needed to distribute approvals to these groups so that users of all lower clearance level groups can derive k_{OU}. In this case, the users of G_1 need to store a maximum of n private information, while the users of G_n need to store one private information to derive k_{OU}.

3) KEY DERIVATION COST

Let K be the secret key of OU and derived from users' private information to access data. We will address the cost of deriving the key for a user in five categories: the required number multiplication M, division D, subtraction S, addition A, and modular mod operations.

In our proposed scheme, the key derivation involves constructing interpolating polynomial and finding a value of the polynomial. Newton's interpolation method is utilized to process the interpolation, and Horner's rule is applied to derive a secret key by performing the polynomial operation more efficiently, and at minimal cost [44].

Assume that set of (t_i, w_i) threshold schemes defined for C_k^{γ} is $\bigcup_{i=k}^{l}(t_i, w_i)$. Considering the worst-case scenario, the actual key derivation cost depends on the maximum threshold level of t_{max} (See 2^{nd} **if** statement of Algorithm 2). Although addition operations A of multiple key components c_i are needed to derive the K, the construction cost of c_i with the highest threshold level t_i will be decisive for the total cost.

According to Knuth [44], Newton interpolation method needs $(d^2+d)S$ and $(d^2+d)/2D$ operations for a polynomial of degree d. In order to derive the key component $(c_i = p_i(0))$ from a polynomial p_i, $(2d-1)M$ multiplication, $(2d)A$ addition and 1 modular operations are needed. Note that d is the degree of the interpolation polynomial and threshold level of t_{max} is equal to $d+1$.

The total cost of the key derivation phase involving both the newton interpolation method and Horner's rule [44] is

$$(d^2+d)S + \frac{d^2+d}{2}D + (2d-1)M + (2d)A + 1mod.$$

For simplicity, the ultimate key derivation cost for each user is bounded by $O(d^2.\log p)$ bit operations where p is a prime number used to construct a prime field $\mathbb{F}_p$.

B. SECURITY ANALYSIS

1) KEY RECOVERY SECURITY (KR_s)

The main goal of almost all hierarchical key access control schemes is to guarantee at least KR_s, which means the k_{OU_1} of G_1 has to be protected against collaboration attacks from any unauthorized or insufficiently approved users. u Our scheme that provides robust key access control is based on Shamir's secret sharing algorithm using a (k, n) threshold scheme that provides information-theoretic security. From the security perspective, if any user has $k-1$ or fewer shares, k_{OU_1} cannot be derived, and no information about k_{OU_1} can be extracted.

TABLE 2. Comparison with other schemes.

Schemes	Dynamic	Private Storage	Public Storage	Key Derivation	Security Type	Security Assertion
Akl and Taylor based [10] [11] – [18], [20], [21]	No	One	$n.T$	T_{exp}	KR_s	N/A
Ateniese et al. TLEBC [22]	Yes	One	$n^2.T^3$ (max)	dt_{se}	KI_s	CPA_s
Ateniese et al. TLPBC [22]	Yes	T (max)	n^2 (max)	dt_p	KI_s	CPA_s
Freire et al. PRF-based [29]	No	w	Zero	$h.T_{prf}$	SKI_s	PRF
Freire et al. FSPRG-based [29]	No	w	Zero	$h.T_{prg}$	SKI_s	$FSPRG$
Tang et al. [38]	Yes	Four	n^2+1 (min)	$2A$+$4M$	SKI_s	PRF
Ours	Yes	n (max) One (min)	Zero	$d^2.\log p$	KI_s	$TSBITS$

Notation:
n: number of classes in the hierarchy
p: a prime number used to construct a prime field F_p
T : total number of time period
T_{exp}: exponential computational time over a large group
KR_s : key recovery secure
dt_{se} : decryption time using a symmetric key encryption scheme
KI_s : key indistinguishability secure
CPA_s : secure against chosen plaintext attack
dt_p : decryption time using pairing evaluation
w : partially ordered set (poset) width
h : path length between the classes in hierarchy
SKI_s : Strong key indistinguishability secure
T_{prf} : time of pseudo-random functions (PRF) evaluation
T_{prg} : time of pseudo-random generator (PRG) evaluation
$FSPRG$: forward-secure pseudorandom generator
A : computational time of modular addition over F_q
M : computational time of modular multiplication over F_q
d : the degree of the interpolation polynomial
$TSBITS$: threshold scheme-based information-theoretic secure

Any user in any class in the hierarchy has to obtain enough approvals by the same level or higher security clearance levels to derive the k_{OU_1} of OU_1 following the security policy defined by the data owner.

According to the example in Table 1, a user in G_2 needs to construct $\{c_2, c_1\}$. Based on the second option chosen by the data owner, which means that the user needs six different shares like (x_i, y_i) belonging to any user in G_2 and also one more share belonging to any user in G_1. Let's assume that there is enough share (only 1) to construct c_1, but only 5 shares from the members of G_2. In such a case, it is computationally impossible to construct c_2 as it requires correctly guessing a point on the graph of a random polynomial.

2) KEY INDISTINGUISHABILITY SECURITY (KI_S)

According to the KI_s principle, the attacker should not distinguish between the k_{OU_1} and a random string of the same length which means the state of KI_s. Since the relevant shares stored by users are chosen randomly, the security of our scheme has also reached KI_s.

C. COMPARISON WITH OTHER SCHEMES

Table 2, created by taking various parameters into account, shows the comparison of our scheme with others in the literature.

In Akl and Taylor based schemes [10], [11]–[18], [20], [21], key updating and key derivation are costly that these schemes are impractical for organizations with large number of groups. The methods have no proof of security, and they are not collusion-resistant. Moreover, only KR_s and KI_s hold under the RSA assumption or random exponentiation. A modified version which is provably secure and KR_s and KI_s holds first, comes up with [27].

In Ateniese *et al.* [22], key derivation cost is tolerable regardless of path length between classes in the hierarchical structure, but the main disadvantage is related to storage need. For the first proposed scheme, both the number of classes and the number of time periods are parameters that define the public storage need. For the second proposed scheme, the number of time periods is critical for identifying the private storage need.

In Tang *et al.* [38], without the need for iterative computation, the direct access scheme handles the possible changes in the hierarchy with light computations and also obtains SKI_s. However, the main disadvantage is the amount of public storage need compared to others, and there is a trade-off between computation cost and storage. Each class needs to compute two times on M, and once on A over F_q to derive its K. On the other hand, each class needs to compute the value of F two times, four times the value of M (modular multiplication time over F_q), and two times the value of A (modular addition time over F_q). Therefore, the overhead of each class is not efficient and tolerable. Also, in case of a change in hierarchy, the data owner must compute and publish a new public matrix. Another disadvantage is that the matrix must be square to establish the relationship between the public storage and the number of classes in the hierarchy, especially on the rekeying.

While our scheme's private storage need is the maximum number of classes n in the OU, this need can decrease to 1. The public storage need is zero in our method. In addition, the key derivation cost of our scheme providing KI_s is $d^2 . \log p$.

V. CONCLUSION

In this work, Shamir's secret sharing scheme and Newton's interpolation method have been exploited to construct a flexible hierarchical key access control mechanism that can be employed in various real-time scenes, especially for any cloud infrastructures. Public and private storage needs are one of the main overheads for the data owners. Our scheme has reduced the concern on the security of data access policy based on a hierarchical structure. The proposed key access control scheme provides a computationally efficient method for key derivation. The scheme is collusion resistant, and this means KR_s and KI_s.

The proposed scheme provides both the private cloud security and the functionality, accessibility, and cost savings of the public cloud. With the use of the public cloud by companies, other advantages such as the reliability of the public cloud and the minimum maintenance and management requirements are obtained.

The other advantages are as follows:

1) The control of data can be provided entirely by the data owner,
2) It provides hierarchic and organization unit-based management of data to be processed on the public cloud,
3) Organization unit and a user group based security level policies, namely the multi-hierarchical security mechanism is provided,
4) Security level policies of the company are dynamically adjustable by the data owner,
5) The user can be a user group member for any organizational unit by the data owner. Thus, the user has authority over the data of another organizational unit.

REFERENCES

[1] E. Thomas, P. Ricardo, and M. Zaigham, *Cloud Computing: Concepts, Technology, and Architecture*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2013.
[2] M. Peter and G. Timothy, "The NIST definition of cloud computing, recommendations of the national institute of standards and technology," NIST, Gaithersburg, MD, USA, Tech. Rep. 800-145, 2013.
[3] *Information Technology-Cloud Computing-Reference Architecture*, Standard ISO/IEC 17789:2014, 2014.
[4] *Information Technology-Cloud Computing-Overview and Vocabulary*, Standard ISO/IEC 17788:2014, 2014.
[5] L. Zhou, V. Varadharajan, and M. Hitchens, "Achieving secure role-based access control on encrypted data in cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 12, pp. 1947–1960, Dec. 2013.
[6] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, Tenerife, Spain, 2010, pp. 136–149, doi: 10.1007/978-3-642-14992-4_13.
[7] X. Dong, J. Yu, Y. Luo, Y. Chen, G. Xue, and M. Li, "Achieving secure and efficient data collaboration in cloud computing," in *Proc. IEEE/ACM 21st Int. Symp. Qual. Service (IWQoS)*, Jun. 2013, pp. 1–6, doi: 10.1109/IWQoS.2013.6550281.
[8] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.
[9] L. Zhou, V. Varadharajan, and M. Hitchens, "Trust enhanced cryptographic role-based access control for secure cloud data storage," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 11, pp. 2381–2395, Nov. 2015.
[10] S. G. Akl and P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," *ACM Trans. Comput. Syst.*, vol. 1, no. 3, pp. 239–248, 1983.
[11] S. J. Mackinnon, P. D. Taylor, H. Meijer, and S. G. Akl, "An optimal algorithm for assigning cryptographic keys to control access in a hierarchy," *IEEE Trans. Comput.*, vol. C-34, no. 9, pp. 797–802, Sep. 1985.
[12] R. S. Sandhu, "Cryptographic implementation of a tree hierarchy for access control," *Inf. Process. Lett.*, vol. 27, no. 2, pp. 95–98, 1988.
[13] L. Harn and H.-Y. Lin, "A cryptographic key generation scheme for multilevel data security," *Comput. Secur.*, vol. 9, no. 6, pp. 539–546, Oct. 1990.
[14] C.-C. Chang, R.-J. Hwang, and T.-C. Wu, "Cryptographic key assignment scheme for access control in a hierarchy," *Inf. Syst.*, vol. 17, no. 3, pp. 243–247, May 1992.
[15] H. T. Liaw, S. J. Wang, and C. L. Lei, "A dynamic cryptographic key assignment scheme in a tree structure," *Comput. Math. Appl.*, vol. 25, no. 6, pp. 109–114, Mar. 1993.
[16] M. S. Hwang, C. C. Chang, and W. P. Yang, "Modified Chang-Hwang-Wu access control scheme," *Electron. Lett.*, vol. 29, no. 24, pp. 2095–2096, 1993.
[17] H.-T. Liaw and C.-L. Lei, "An optimal algorithm to assign cryptographic keys in a tree structure for access control," *BIT*, vol. 33, no. 1, pp. 46–56, Mar. 1993.
[18] H. Min-Shiang, "A cryptographic key assignment scheme in a hierarchy for access control," *Math. Comput. Model.*, vol. 26, no. 2, pp. 27–31, Jul. 1997.
[19] P. D'Arco, A. De Santis, A. L. Ferrara, and B. Masucci, "Variations on a theme by Akl and Taylor: Security and tradeoffs," *Theor. Comput. Sci.*, vol. 411, no. 1, pp. 213–227, 2010.
[20] W.-G. Tzeng, "A time-bound cryptographic key assignment scheme for access control in a hierarchy," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 1, pp. 182–188, Aug. 2002.
[21] H.-Y. Chen, "Efficient time-bound hierarchical key assignment scheme," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 10, pp. 1301–1304, Oct. 2004, doi: 10.1109/TKDE.2004.59.
[22] G. Ateniese, A. De Santis, A. L. Ferrara, and B. Masucci, "Provably-secure time-bound hierarchical key assignment schemes," *J. Cryptol.*, vol. 25, no. 2, pp. 243–270, 2012, doi: 10.1007/s00145-010-9094-6.
[23] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.
[24] X. Yi, "Security of Chien's efficient time-bound hierarchical key assignment scheme," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 9, pp. 1298–1299, Sep. 2005, doi: 10.1109/TKDE.2005.152.
[25] X. Yi and Y. Ye, "Security of Tzeng's time-bound key assignment scheme for access control in a hierarchy," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 4, pp. 1054–1055, Jul./Aug. 2003, doi: 10.1109/TKDE.2003.1209023.
[26] C. Jason, M. Keith, and W. Peter, "On key assignment for hierarchical access control," in *Proc. Comput. Secur. Found. Workshop*, 2006, pp. 98–111, doi: 10.1109/CSFW.2006.20.
[27] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, "Dynamic and efficient key management for access hierarchies," *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 3, pp. 1–43, Jan. 2009.
[28] V. R. L. Shen and T.-S. Chen, "A novel key management scheme based on discrete logarithms and polynomial interpolations," *Comput. Secur.*, vol. 21, no. 2, pp. 164–171, 2002.
[29] E. S. V. Freire, K. G. Paterson, and B. Poettering, "Simple, efficient and strongly KI-secure hierarchical key assignment schemes," in *Topics in Cryptology—CT-RSA* (Lecture Notes in Computer Science), vol. 7779, E. Dawson, Ed. Berlin, Germany: Springer, 2013.
[30] E. Bertino, N. Shang, and S. S. Wagstaff, Jr., "An efficient time-bound hierarchical key management scheme for secure broadcasting," *IEEE Trans. Dependable Secure Comput.*, vol. 5, no. 2, pp. 65–70, Apr./Jun. 2008, doi: 10.1109/TDSC.2007.70241.
[31] Y. F. Chung, H. H. Lee, F. Lai, and T. S. Chen, "Access control in user hierarchy based on elliptic curve cryptosystem," *Inf. Sci.*, vol. 178, no. 1, pp. 230–243, Jan. 2008.
[32] F.-G. Jeng and C.-M. Wang, "An efficient key-management scheme for hierarchical access control based on elliptic curve cryptosystem," *J. Syst. Softw.*, vol. 79, no. 8, pp. 1161–1167, Aug. 2006.
[33] H. M. Sun, K. H. Wang, and C. M. Chen, "On the security of an efficient time-bound hierarchical key management scheme," *IEEE Trans. Dependable Secure Comput.*, vol. 6, no. 2, pp. 159–160, Apr. 2009.

[34] A. K. Das, N. R. Paul, and L. Tripathy, "Cryptanalysis and improvement of an access control in user hierarchy based on elliptic curve cryptosystem," *Inf. Sci.*, vol. 209, pp. 80–92, Nov. 2012.
[35] Y.-L. Lin and C.-L. Hsu, "Secure key management scheme for dynamic hierarchical access control based on ECC," *J. Syst. Softw.*, vol. 84, no. 4, pp. 679–685, 2011.
[36] A. De Santis, A. L. Ferrara, and B. Masucci, "Efficient provably-secure hierarchical key assignment schemes," *Theor. Comput. Sci.*, vol. 412, no. 41, pp. 5684–5699, 2011.
[37] H. R. Hassen, H. Bettahar, A. Bouadbdallah, and Y. Challal, "An efficient key management scheme for content access control for linear hierarchies," *Comput. Netw.*, vol. 56, no. 8, pp. 2107–2118, May 2012.
[38] S. Tang, X. Li, X. Huang, Y. Xiang, and L. Xu, "Achieving simple, secure and efficient hierarchical access control in cloud computing," *IEEE Trans. Comput.*, vol. 65, no. 7, pp. 2325–2331, Jul. 2016.
[39] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford InfoLab, Stanford, CA, USA, Tech. Rep. 1999-66.
[40] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, Savannah, GA, USA, 2016, pp. 265–283.
[41] C.-L. Hsu and T.-S. Wu, "Cryptanalyses and improvements of two cryptographic key assignment schemes for dynamic access control in a user hierarchy," *Comput. Secur.*, vol. 22, no. 5, pp. 453–456, Jul. 2003.
[42] S. Zhong and T. Lin, "A comment on the Chen-Chung scheme for hierarchical access control," *Comput. Secur.*, vol. 22, no. 5, pp. 450–452, Jul. 2003.
[43] D. Tiwari and G. R. Gangadharan, "SecCloudSharing: Secure data sharing in public cloud using ciphertext-policy attribute-based proxy re-encryption with revocation," *Int. J. Commun. Syst.*, vol. 31, no. 5, p. e3494, Mar. 2018.
[44] D. E. Knuth, *The Art of Computer Programming*, 3rd ed. Reading, MA, USA: Addison-Wesley, 1998.

BARIS CELIKTAS received the B.S. degree in system engineering from Turkish Military Academy, Turkey, in 2008, the M.S. degree in international relations from Karadeniz Technical University, in 2016, and the M.S. degree in applied informatics from Istanbul Technical University, in 2018, where he is currently pursuing the Ph.D. degree in cyber security engineering and cryptography with the Institute of Informatics, Department of Applied Informatics. He is currently working as an IT System and Security Manager with NATO. His current research interests include cyber security, network security, cloud computing, and cryptography.

IBRAHIM CELIKBILEK received the B.S. degree in electronics and computer from Fırat University, Turkey, in 2002, and the M.S. degree in cybersecurity engineering from Istanbul Sehir University, in 2016. He is currently pursuing the Ph.D. degree in information and communications engineering with the Institute of Informatics, Department of Applied Informatics, Istanbul Technical University, Istanbul, Turkey. His current research interests include computational number theory, lattice-based cryptography, post-quantum cryptography.

ENVER OZDEMIR (Member, IEEE) received the B.S. degree in mathematics from Middle East Technical University, Turkey, in 2002, and the Ph.D. degree in mathematics from The University of Maryland, College Park, USA, in 2009. He was a Research Fellow with CCRG, Nanyang Technological University (NTU), Singapore, from 2010 to 2014. He is currently an Associate Professor with the Informatics Institute, Istanbul Technical University and he is also the Deputy Director of the National Center for High-Performance Computing (UHeM). His current research interests include cryptography, computational number theory, and network security.