

An Inner Product Space-Based Hierarchical Key Assignment Scheme for Access Control

BARIS CELIKTAS¹, SUEDA GUZEY¹, AND ENVER OZDEMIR¹, (Senior Member, IEEE)

¹Cyber Security Engineering and Cryptography Department, Institute of Informatics, Istanbul Technical University, Istanbul, Turkey

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

ABSTRACT An inner product space-based hierarchical key assignment/access control scheme is presented in this work. The proposed scheme can be utilized in any cloud delivery model where the data controller implements a hierarchical access control policy. In other words, the scheme adjusts any hierarchical access control policy to a digital medium. The scheme is based on inner product spaces and the method of orthogonal projection. While distributing a basis for each class by the data controller, the left-to-right and bottom-up policy can provide much more flexibility and efficiency, especially during any dynamic update/change in the hierarchical structure. For each class, the secret keys can be derived only when a predetermined subspace is available. A parent class can obtain the secret keys of its child classes, which means a one-way function, and the opposite direction is not allowed. Our scheme is collusion attack and privilege creep problem resistant, as well as key recovery and indistinguishability secure. The performance analysis shows that the data storage overhead is more tolerable than other schemes in the literature. In addition, the other advantage of our scheme over many others in the literature is that it needs only one operation for the derivation of the secret key of child classes.

INDEX TERMS access control, hierarchical, inner product, key assignment, vector space.

I. INTRODUCTION

THE confidentiality of data in the digital medium is provided via cryptographic primitives such as a symmetric-key algorithm. An encrypted data in any place is converted to its original form with a predetermined secret key. The access control policy for the predetermined key is supposed to reflect the data controller's policy which might be complicated in certain organizations. The key extracting process in most cases involves the approval of more than one user and an efficient application of any secret sharing algorithm [1] handles the desired key access control in some cases. On the other hand, a key access control policy that requires approvals from various users where each one has a distinct classification level determined by the data controller may not be practically adapted from a secret sharing algorithm. Since Akl and Taylor's proposed hierarchical access control scheme in 1983 [2], many studies have appeared on hierarchical key assignment schemes. However, what motivates us to do this research is that there are still open avenue to be completed towards a practical access mechanism for hierarchical structures.

The recent trend of moving various services to a digital medium is welcomed by many organizations that process mission-critical data. Such organizations might prefer to use public or private cloud services for data flow and apply their data access policy to the data in such a digital environment. There are many concerns about using the public cloud, especially for military, health, and banking, where confidentiality and privacy are crucial. Besides the general concerns of confidentiality, availability, integrity, reliability, data lock-in, and regulatory compliance, integration of the data controller's access policy to a digital medium stands as a challenging topic in the research community [3], [4]. Due to the concerns mentioned above, many organizations are slowing down their digitalization adaption plans even though the public cloud deployment model provides many advantages, especially in total time-cost [5].

In this work, we present a practical key access policy that securely reflects data controllers' hierarchical access policy to eliminate the hesitation in the adaption of access policy to the public cloud. The resulting scheme might help to alleviate concerns about moving private data to the public cloud. The

scheme allows flexible hierarchical key assignment protocol which is based on inner product spaces utilizing a mathematical tool of orthogonal projection (OP). The scheme is specially designed for organizations that use storage as a service cloud delivery model from the public cloud. This work considers Bell-LaPadula's (BLP) hierarchical multi-level lattice-based model that addresses confidentiality in a hierarchical organizational structure similar to government or military organizations. One of the main properties that should address is simple security, which means any class with lower classification privileges cannot read or access an object of a higher classification. Thus, we follow the read-down model for hierarchical access control. Figure 1 illustrates a multilevel hierarchical organizational structure. The number of levels indicates the number of classification levels defined by the data controller.

Various organizations have hierarchical management mechanisms in their organizational structure. The hierarchical mechanism can be designed by dividing the organization into functional areas, that is, by dividing it into smaller organization units/groups which is denoted by G_i . Each G_i has a security classification level defined by the data controller to access the key/data. We are assuming that only members of higher classification levels can access data of lower classification levels, if there is a parent-child relationship. Since the access control is managed by the data controller and on-premise, a cloud storage entity as a service provider should not be able to obtain any information about the user's mission-critical data. Thus, a scheme should facilitate the public cloud and be employed securely for various purposes. The first stage of such a scheme that addresses the users' key access control problem is presented in this work.

In the following parts, we employ directed acyclic graphs (DAG), called access graph and denoted by $G(V, E)$ where V is a finite set of classes (vertices) and E is a set of paired vertices (edges). Let V be a collection of sets $\{V_1, V_2, \dots, V_i\}$. For example, the graph $G(9, 8)$ in Figure 1 is a DAG with nine vertices and eight directed edges. \leq denotes a partial order (binary relation - antisymmetric, transitive, and reflexive) on V , thus (V, \leq) is a partially ordered set (poset). Security classes V_i , in other words, the groups G_i 's are disjoint. Let $x, y \in V$, $x \leq y$ indicates two disjoint classes and it means that the users in class x can access all the data to which the users in class y have access. Namely, $G_x \leq G_y$ means that any user in G_x can access the data belongs to G_y , and G_x 's classification level is higher than or equal to G_y . Note that G_x has to be one of the parent classes of G_y . Figure 1 and Figure 2 illustrate the key access structure for the proposed scheme. In this figure, the root parent G_1 has two children (G_2, G_3). Each of these children is a parent to other children. Parent G_2 has three children (G_4, G_5, G_6), parent G_3 has also three children (G_7, G_8, G_9). Each child has only one immediate parent, all parents except the root parent G_1 also have an immediate parent and children. The secret key of each child class can be obtained by the parent classes and by its own members. Although two classes might have the

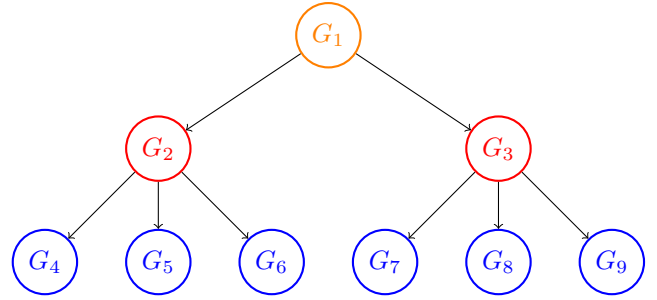


FIGURE 1. Left to right and bottom to up policy-based partially ordered set (or poset) in hierarchy.

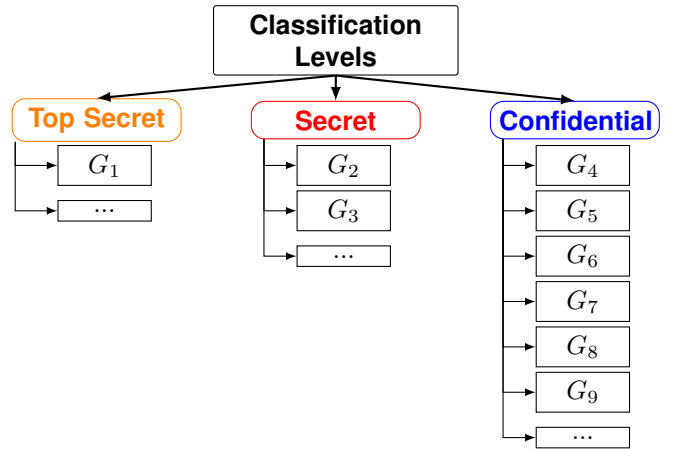


FIGURE 2. Taxonomic ranks of classification levels.

same classification level (as shown in Figure 2), if they are not in the parent-child relationship in Figure 1, the key of a class should not be obtained by the other class.

The remaining paper is structured as follows; Section II presents a literature review regarding key assignment schemes for hierarchical access control. Section III provides the preliminaries on which our scheme is based, such as the inner product space, the Gram-Schmidt (GS) method, and the OP. Section IV presents the detailed steps of the proposed scheme including basic rules, system-ready, key derivation, and dynamic update phases. Section V devoted for the experiment results including performance, security, and comparison analyses of our scheme. Section VI provides a brief summary with conclusion.

II. LITERATURE REVIEW

This section will present the studies related to the key assignment/access control schemes for hierarchical infrastructures.

In order to provide a key assignment scheme that conforms to a hierarchical structure, various schemes have been presented, and almost all the schemes named Akl and Taylor-based schemes [2], [6] – [13] are based on a poset hierarchy. They are not designed to give access to users only for a certain period of time. The other point that is overlooked is that updating keys is based on poset hierarchy. In other words, the key derivation might be so costly that they become

not be practical for large hierarchies. In addition, the work [14] provides an analysis of the Akl-Taylor scheme according to the definitions proposed by [15]. It also shows a method to choose the public parameters in order to get a variation of the scheme that is provably-secure against key recovery under the RSA assumption.

In the work [16], the Lucas function-based time-bound property is utilized to ensure time and performance efficiency and to solve key update issues. These time-bound related schemes are divided into two categories: the first is based on tamper-proofed machines [17] and the second [16], [18] is based on public information. Note that tamper-proof machines are collusion (collaboration attack) resistant but expensive, inappropriate for the cloud, and limit user convenience. Public information can be utilized efficiently since the cloud ensures broad network access. Both public information and the user's own key are used to derive the secret key of lower classification levels. Thus, the number of public information is critical for measuring the efficiency of the key assignment scheme.

In all other schemes proposed later, the main goal is to ensure secure, dynamic, and cost-effective hierarchical key assignment method. The parameters that gauge the efficiency and security of the hierarchical key assignment schemes in the literature are as follows:

- 1) **The amount of public and private information needed:** It refers to the amount of information allocated to each class in the hierarchy for the secret key derivation. The data controller distributes private information to classes so that only users of desired classes have access. On the other hand, the data controller publishes all public information to all classes following the predetermined hierarchical structure.
- 2) **Key derivation complexity:** It refers to the number of operations or computational requirements/cost needed for key derivation and it must be minimal and tolerable.
- 3) **Key update complexity/cost due to the changes in the hierarchy:** It refers to the cost needed for key updates and it must be minimal and tolerable. A scheme is supposed to allow the insertion and deletion of classes in the organization without the need for redistribution of any private information.
- 4) **Collusion (aka collaborative/key recovery- KR) attack resistance:** The derivation of the secret key of any class must be secure against any collusion of users of child classes, and key recovery secure KR_s denotes such a secure scheme.
- 5) **The key indistinguishability secure denoted by KI_s :** It should not be possible to distinguish between an arbitrary string and the secret key of the same length. If the scheme is KI_s , then it is also KR_s , but not vice versa [18].
- 6) **Resistance to privilege creep problem:** When the classification level of any member U of the class is downgraded, or the membership is changed, the member should not be able to use former privileges during

a period of transition. In other words, a scheme must guarantee both forward and backward secrecy.

The work [2] proposed a key management (access control) scheme to solve the multi-group management and the problem of data sharing in the hierarchical structure. In this scheme, the communication (or computer) system users are classified into disjoint sets $G_1, G_2, G_3, \dots, G_n$. Note that the relationship between classes is just a poset hierarchy. A security level is used to define each of the G_n and users in G_n can access data kept by users in the same group or lower security level group, while the opposite direction is not allowed. They offer a solution to a hierarchical access control problem. The approach does not entirely address multi-level security problems although it is useful in a distributed and secure system. It cannot be adapted dynamically and flexibly to the security policy set by the data controller. The other problem is that users can use the key permanently at a higher security level. A significant amount of storage and communication is consumed due to the need to replace the keys regularly and redistribute them to users. According to [18], the scheme derives the key expensively and is only KR_s . Because the number of keys kept by each user is large, it is not cost-effective as the number of users rises [6]. A significant amount of storage is also needed to store public information for each security class [8], [13].

The work [6] which is the improved scheme of [2] (aka a canonical assignment) is presented to mitigate the public information storage need in order to control data access within a user group ordered in a hierarchical structure if the number of classes is remarkably huge [16]. But, the storage need is not zeroed [13]. Any user of a group can access the data of lower-level group as they can derive lower-level group users' keys and it is also KI_s .

The work [7] proposed a tree hierarchical scheme based on symmetric-key cryptography that security classes are designed as rooted-tree, an example of poset hierarchy. It uses an iterative method of a one-way function, which provides an efficient method to compute images of inputs while making computationally hard to compute pre-images. With utilizing such functions, the key of the child class can be derived. The most important innovation is that novel security classes can be inserted even the keys of existing classes remain unchanged. For example, once a new security class is inserted in [2] and [6], all keys not associated with this class also have to be changed, and this causes a substantial burden, especially for distributed and large infrastructures. In addition, the key derivation process does not need additional public parameters. But, the main drawbacks of it are as follows. The computational overhead during deriving keys, particularly once the key belongs to the lowest class that needs to be created by the root of the tree, is not tolerable. It can only be implemented efficiently in case of less than eleven security level-tree hierarchy.

The work [8] proposed a method akin to [2] but adopted a bottom-up key derivation policy. Unlike other works [2], [6], new security classes can be inserted even the keys of

existing classes remain unchanged. The public storage need for classes is much less than [2], [6]. In addition, the scheme is more efficient in memory utilization in comparison with [2].

The works [9] and [10] proposed schemes based on Newton's implementation and one-way function. However, the computation time needed for key generation and derivation is massive, which makes them time-consuming. In addition, their schemes are insecure against collaboration attacks [11].

The work [12] proposed an optimal heuristic algorithm for distributing keys applying an up-bottom design approach in a tree hierarchy. The generation and derivation of keys can be done efficiently, as well as the method reduces the storage requirement for general parameters. However, similar to [7], it can only be utilized in a tree hierarchy.

The work [13] modified the algorithm [12] so that it can be used in the poset hierarchy. Thus a user at a parent class can derive the keys of users of child levels using its own cryptographic key, which means a one-way function, and the opposite direction is not allowed. The users collaborating at a child class of the hierarchy would not derive the key of a parent class because they are not authorized. Thus, it is KR_S .

The work [16] is based on a time-bound and inspired by [2] to hinder the key from being utilized continuously by higher level users of class C . In this scheme, any user can become a member of C only for a certain time period. A user in C_x can only derive K_y from K_x at time t iff $C_y \leq C_x$ and $t_1 \leq t \leq t_2$, where t_1 and t_2 are the boundaries/limits of the time period. Namely, a user can hold encrypted data for only within a certain time period. A hierarchy with optimal bandwidth has broadcast data to authorized users, and a user can only receive data that has been granted access. On the other hand, by listening to the broadcast, any data cannot be obtained by unauthorized users. A user of the parent class can grant privilege to another user to decrypt encrypted data which provides flexibility. It is not dependent on the number of classes in the hierarchical structure, which does not exist in the previously proposed key assignment schemes based on poset hierarchy. Since users must always hold keys to access authorized data for a certain time period, it is not efficient as expected. While it requires less communication and storage cost, it is computationally inefficient due to the need for costly public key computation in addition to costly computations which occur overload during the implementation [17]. In addition, it is not KI_s if at least three users collude to access the keys [20].

The work [17] is motivated especially by [2] and [16] for its efficient time-bound feature. It improves [16]'s scheme by allocating different keys to all in hierarchical structure to address both implementation and performance problems. It uses a tamper-resistant machine performing only simple operations without public-key encryption and is inaccessible even to the controller. There is a Trusted Agent (TA) and a secure one-way hash function h as well. It is economically unfeasible to derive a key from the public value. Users of child classes cannot derive the key of parent classes, meaning

KR_s . No user can derive any key beyond the authorized time periods. In comparison with [16], it appears much more efficient based on performance analysis. It requires a low-cost, tamper-resistant machine with low computational complexity supporting small storage. However, three or more users' collaboration is enough to access the keys, so it is not KR_s [19], [20].

The work [21] categorized nearly all key assignment schemes in literature as a trivial key assignment scheme-TKAS, a trivial key encrypting key assignment scheme-TKEKAS, a direct key encrypting key assignment scheme-DKEKAS, a node-based key assignment scheme-NBKAS, and an iterative key encrypting key assignment scheme-IKEKAS. In TKAS, key generation is effortless, but it is a poor scheme since the key update/change is arduous. In TKEKAS, the key update process is easier and useful, especially if the key is compromised. In DKEKAS, private storage need is minimal, and key update is easier, but public data is quite high. In NBKAS, whose security based on the difficulty of computing integral roots modulo n , storing a single secret value is only required for each user. It has an advantage over both TKEKAS and DKEKAS. The keys are originally dependent but can easily be converted to independent keys. Keys can be derived in a single step for the schemes above. In IKEKAS, key generation and updates are relatively easy, less public storage is needed, but the key generation is iterative, not direct. It is stated that the changes in the information flow policy are included in a key assignment scheme, but there is a need for studies that address the key update/change problems. In addition, best practices for any key assignment scheme are described as follows: requiring a small amount of private and public storage, providing a computationally efficient method for both derivation and update keys, and being KR_s (no combination of users can derive keys where they do not have authorization).

The work [15] proposed a dynamic and efficient key assignment/access control scheme for hierarchical infrastructures. It is based on random access graphs. In order to derive the child's key using its own key, only the hash functions are used for a node. The derivation of child class's key is restricted to the number of linear bit operations, and each class has a single key related to that class. KI_s and KR_s first took their place on the stage with [15]. This scheme's security depends on pseudo-random functions and an employed symmetric key encryption scheme. Indeed, if the encryption scheme is secure under a chosen plaintext attack, then the scheme is KI_s . In the worst case, the key derivation needs the implementation of $O(n)$ hash functions, where n is the number of nodes in the graph. It is more efficient than its predecessors due to its dependency on interpolating polynomial, costly modular exponentiation operations, and additional encryption. However, according to [18], each user must store up to three private information, and the overall amount of public information adverse affects the key derivation complexity. In [23], the greater the public information, the larger the number of both edges and classes on the graph.

Also, the larger the number of classification levels between classes, the greater the key derivation time-cost. Similar to [15], the scheme in the work [22] is also suitable to the dynamic changes of classes in the structure, such as deletion and insertion of classes.

In order to address hierarchical access control problem, key assignment schemes [24] – [26] based on elliptic-curve cryptography (ECC) were proposed. In [24], the principle that the number of access control policies is related to the number of keys and tamper-resistant machines performing a major role in this scheme, makes it slower than both [16] and [17]. According to the work [27], the scheme [24] is not KR_s and the scheme [25] is not secure against exterior root-finding attack. Furthermore, the method presented in [26] is not KR_s [29].

The schemes [30] and [31] provide better performance compared to their predecessors. In [30], the schemes named time-bound and time-bound broadcast encryption-based family (respectively TBEBF and TBEBF) provide dynamic updates without the need for redistribution of private information despite the rise in the number of public information. They are KI_s and improve Atallah et al.'s scheme [15] on the computational need for key derivation and update. Key derivation (TBEBF) requires symmetric decryption about as many as the number of levels in the hierarchical structure, and TBEBF requires complex symmetric decryption. The private storage need is small as it needs only one key per class, but the greater the public storage need, the larger the number of edges and classes in the graph. However, in [31], it guarantees efficient computation and storage cost compared to its predecessors. The schemes [18] are based on symmetric encryption and bilinear maps respectively. They are KI_s and KR_s , meaning provable-secure. The key derivation operations are very efficient as only one decryption, or one pairing evaluation is sufficient regardless of the number of levels in the hierarchical structure. Only changes to the public information are sufficient to update the structure and any private information remains unchanged. According to the work [23], since they are based on the number of both time periods and classes, private information can be as huge as the number of time periods.

The schemes [23] are based on forward secure pseudo-random generators (FSPRG) and pseudo-random functions (PRF). There is a trade-off between the key derivation efficiency and private information storage need. The overall key derivation efficiency relies on the longest poset depth. In contrast, private information need relies on the poset width, but key derivation efficiency is comparatively better than others. Another advantage is that the schemes don't need public storage. In addition, the updated version of [15] guarantees a stronger security SKI_s than all schemes before.

The scheme [32] is based on linear-geometry, organizes the set of users in a hierarchy and divides the users into security classes (disjoint groups) to ensure different access privileges for each. To derive the key of a security class, its own public vector and the private vector of its parent class

must be used together. The key of the child security class can be derived directly by the parent security class, without the need for iterative computation and it is also SKI_s . It requires to calculate the vector multiplication and the pseudo-random function values, which causes tolerable computational cost. While the public storage need is greater than the others, which is the main drawback of this scheme, there is a trade-off between the storage need and the computation cost. It also provides fine-grained control and flexibility to adapt to changes/updates in the hierarchy efficiently. If there is a change, a new public matrix must be calculated and published by the data controller. Thus, the overall cost might not be tolerable and efficient eventually. The other disadvantage is that the matrix should satisfy certain properties to set the relation between the number of classes in the hierarchical structure and the public information, particularly for re-keying.

The scheme [33] relies on a secret sharing algorithm and polynomial interpolation method. In this scheme, the data controller is an entity composed of organizational units, the key is secret information that is only one for each organizational unit in the structure. The scheme is especially designed for the dependent organizational units [34]. In other words, the users with sufficient approvals from the higher or same privileged users can only access the key using the topological sorting of a directed graph, including self-looping. The key derivation time-cost, private, and public storage needs are tolerable. Because the secret key of the organization unit is not necessary to be kept anywhere, a data breach relied on the key disclosure risk is also minimized or eliminated.

In summary, overall two types of hierarchical key assignment/access control schemes are proposed, direct and indirect control. For direct schemes such as [23], [30], [31], it only requires one computational task for the derivation of the child class's secret key. For indirect schemes such as [22], [25], [29], [32], any child class's secret key can be derived by computing all the keys in the path to its own class. However, they are not secure enough [28], [38], [39] and require a high overhead task [29]. Therefore, there are still open areas for the research toward a practical and secure key access mechanism. This motivates us to build a secure and efficient inner product space-based cloud-independent hierarchical key assignment scheme.

III. PRELIMINARIES

A. INNER PRODUCT SPACE

Various applications of linear algebra have been presented for the last century [35]. Matrices, eigenvalues, linear systems are indispensable tools in computational science, especially in artificial intelligence and machine learning-related research [36]. The majority of vector spaces come with a well-defined inner product which is basically a tool to measure the distance between two vectors in the space. Such spaces have been recently employed by the security community [37]. Once the distance is defined in space, locating the closest vectors to certain subspaces will only require computational

tasks. The proposed algorithm in this work uses this fundamental notion of distance, in other words, the inner product. An inner product and its properties on a vector space V are briefly described below. Then, the procedure of finding the closest vector to a defined subspace is presented. As the procedure requires utilizing an orthonormal basis for the given subspace, constructing an orthonormal basis for a vector space called the GS orthogonalization process is briefly demonstrated. Finally, we explain the last step to find OP of a vector to the defined subspace of V .

The vector space V with an inner product is called an inner product space. Utilizing an inner product space for our purposes might first require creating an orthonormal basis in it. Now assume $B = \{v_1, v_2, \dots, v_n\}$ is a basis for the vector space V . The GS method can be utilized to make it an orthogonal basis for V , and the new set $S = \{w_1/\sqrt{\langle w_1, w_1 \rangle}, \dots, w_n/\sqrt{\langle w_n, w_n \rangle}\}$ after applying GS method to the set B forms an orthonormal basis for V .

The inner product has been presented to create an analogue of the OP for a vector subspace. An OP can be rephrased as follows. Let g be a vector and W be a subspace of V such that it is generated by g . Assume that the vector $f \in V$ doesn't lie in W . The formula that gives the OP of f on W is $\frac{\langle f, g \rangle}{\langle g, g \rangle} g$. Interestingly, the projection of f on W remains the same for any basis of W , and this fact is independent of the dimension of W , which will be exploited in the proposed key assignment/access control scheme.

B. ORTHOGONAL PROJECTION (OP)

In real-time applications, one needs to approximate a value of a function $f(x)$ where this value can not be computed analytically. For example, Taylor approximation allows one to write down a continuously differentiable function as a linear combination of polynomials. Similarly, Fourier Expansion gives a method to express any periodic function as a linear combination of trigonometric functions. The fundamental idea behind the Fourier Expansion is the well-known OP. Basically, let W be a subspace of a vector space V and f be a vector in V . Assume for a moment that f doesn't lie in W , then OP answers the question of how one finds the closest vector in the subspace W to the vector f . The method indicates that the closest vector h to f is the OP of f onto W . In other words, h is the nearest vector to f in W if and only if $(f - h)$ is perpendicular to all vectors in W . In practice, finding h for a given f is not a tedious task if one knows an orthonormal basis for the subspace W . In fact, if $S = \{g_1, g_2, \dots, g_n\}$ is an orthonormal basis for W then the projection vector of f on W can be written as a linear combination of elements in S . Let h be the closest vector of f then

$$h = c_1 g_1 + c_2 g_2 + \dots + c_n g_n \text{ for some integers } c_i \in \mathcal{F} \quad (1)$$

As $f - h$ must be perpendicular to each one of g_1, g_2, \dots, g_n , the inner product of $f - h$ with g_1, g_2, \dots, g_n must all be

zero. In other words, $\langle f - h, g_i \rangle = 0$ for $i = 1, \dots, n$ and this implies

$$c_i = \langle f, g_i \rangle \text{ for } i = 1, \dots, n. \quad (2)$$

We should note here that for any orthonormal basis S' of W , the OP of f on W will be the same. As long as a basis of W is known, the unique OP can easily be computed via the GS orthogonalization process. Figure 3 depicts the idea of OP on an inner product space.

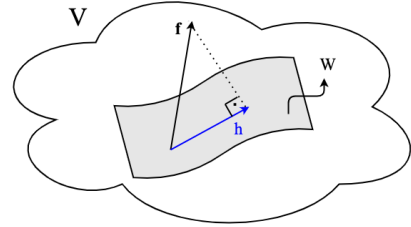


FIGURE 3. Orthogonal Projection: The closest vector h to f is the orthogonal projection of f onto W . In other words, h is the nearest vector to f in W if and only if $(f - h)$ is perpendicular to all vectors in W .

A projection of a vector f onto W is unique and will be a crucial observation that the proposed scheme is used for its purpose. In other words, for a subspace of W , each user has a distinct basis, but each user can construct the same OP of a vector f in V .

IV. THE PROPOSED SCHEME

In this section, we discuss the details of the proposed scheme including basic rules, system-ready, key derivation, and dynamic update phases.

A. BASIC RULES

Our proposed scheme is based on the following rules.

Rule 1. G_1 is the root class and the most privileged group, c is the number of classes in the hierarchical structure where $c \geq 1$.

Rule 2. For any two classes $G_i, G_j \in V$, $i \leq j$ means that any user in G_i can access its own secret key K_i and also K_j belonging to G_j whose classification level is equal to or lower than that of G_i . This is the simple security property (no read-up access control policy) based on BLP hierarchical lattice-based model. Note that if G_i is one of the parent classes but G_i and G_j are not in the parent-child relationship like in Figure 1, then K_j should not be obtained by the members of G_i . However, according to the policy defined by the data controller in our scheme which can run as a directed graph-based topological ordering in the "Dynamic Update Phase" subsection, the K_j can be obtained by the members of G_i .

Rule 3. All basis sets S distributed for each class to design a poset in the hierarchy have to be in compliance with the LRBU policy to provide much more flexibility and efficiency especially for any dynamic update/change in the hierarchy.

For example, the children G_4, G_5 and G_6 of the parent G_2 as depicted in Figure 1 are given the basis sets

$$S_4 = \{v_1, v_2, v_3\}$$

$$S_5 = \{v_4, v_5, v_6\}$$

$$S_6 = \{v_7, v_8, v_9\}$$

for subspaces W_4, W_5 and W_6 respectively.

Rule 4. The union of S_i must also be linearly independent. Note that all v_i for $i = 1, 2, \dots, 9$ are distinct and the set v_1, \dots, v_9 is linearly independent which means that each W_4, W_5 and W_6 are disjoint three-dimensional subspaces, and W_2 is ten-dimensional subspace generated by linearly independent set S_2 containing v_1, v_2, \dots, v_9 and additionally a_1 .

Notice that except v_1, v_2, \dots, v_9 , the remaining vector of S_2 , which is a_1 , is kept secret and it is assumed to be the private information of G_2 which prevents the G_2 's key from being accessible by children G_4, G_5 , and G_6 . In other words, even if all the children collude and combine their private data, they can not generate the subspace W_2 and so they can not derive the secret key of the G_2 . On the other hand, G_2 has all required information (namely all basis elements) to derive its children's secret keys separately to access their data.

Similarly, now consider that G_7, G_8 , and G_9 are children of parent G_3 . Assume that they are given the basis sets

$$S_7 = \{v_{10}, v_{11}, v_{12}, v_{13}\}$$

$$S_8 = \{v_{14}, v_{15}, v_{16}, v_{17}\}$$

$$S_9 = \{v_{18}, v_{19}, v_{20}, v_{21}\}$$

for the distinct four-dimensional subspaces W_7, W_8 , and W_9 respectively. Hence, the higher classification level group G_3 (the parent of these three classes) has the set S_3 containing $v_{10}, v_{11}, \dots, v_{21}$ and another vector a_2 in its basis which generates the subspace W_3 . As mentioned above, except v_{10}, \dots, v_{21} the other element a_2 of the basis for W_3 is kept secret by the G_3 . Thus any user in G_3 can access the corresponding keys K_3, K_7, K_8 , and K_9 but any member of G_4, G_5, G_6, G_7, G_8 , and G_9 which are lower classification level groups does not access the K_3 , they can only access their own secret keys.

Finally, in this scheme, the root class G_1 has the basis S_1 of the vector space W_1 . The basis S_1 consists of the basis elements of G_2 and G_3 as well as its private vector b_1 . Note that, the number of private vector for each parent classes can be flexibly defined by the data controller in advance (e.g. G_2 's private vectors are $\{a_1, \dots, a_i\}$), taking into account that the hierarchical structure can expand horizontally and/or vertically, thus the above illustration can also be adapted efficiently to an N-class hierarchy by applying an LRBu policy.

B. SYSTEM-READY PHASE

The system preparation and distribution phases, in other words, the system-ready phase includes all the steps up to 10th step of Algorithm 1 (See Figure 4).

1) System preparation phase

The data controller:

Step 1. Determines all hierarchical classes G_i for $i = 1, 2, \dots, c$ of the organization displayed in Figure 1 and 2 based on the Rule 2.

Step 2. Determines a vector space V which is preferably infinite dimension over a field \mathbb{F} and decides an inner product defined on V .

Step 3. Identifies subspaces W_1, W_2, \dots, W_c . Note that the selected V structure allows the data controller to change the number of W_i as the number of classes in the hierarchy c changes.

Step 4. Determines the corresponding basis sets S_i according to classification level and dimension policy for predetermined subspaces W_i are distributed to each class/group G_i for $i = 1, 2, \dots, c$ in the organization.

Step 5. Generates the basis sets S'_i obtained by multiplying the first component of the assigned basis set S_i with arbitrary constants in the selected field \mathbb{F} to distribute to users U_i in the class G_i .

2) Distribution phase

At this phase, the data controller shares the public and private information with the relevant classes G_i that enable each member U_i to derive the corresponding key or keys based on the Rule 2.

Step 1. Selects a vector f in V which does not lie in any W_i for $i = 1, 2, \dots, c$ and makes it known by the public.

Step 2. Distributes basis sets S'_i derived from the basis set S_i to each member of the class G_i based on LRBu policy.

C. KEY DERIVATION PHASE

The key derivation phase includes the 11th and 12th steps of Algorithm 1.

Note that once the users receive their own basis, they can form an orthonormal basis for subspaces W_i by applying the GS method to their basis and then apply OP of f onto W_i to derive the corresponding secret key or keys. (See Figure 5).

To derive the corresponding secret key, a user in the hierarchical structure:

Step 1. Applies GS orthogonalization operation to its basis, namely computes corresponding orthonormal basis.

Step 2. Runs OP in order to obtain the corresponding key (See Figure 3). In other words, key derivation process for the user of G_i can be formulated as below:

$$K_{G_i} = \text{Proj}_{W_i} f.$$

Notice that the K_{G_i} for each class G_j requires having a basis for the corresponding subspace W_i . If a child wants to access parent's secret key, it also needs a basis of its

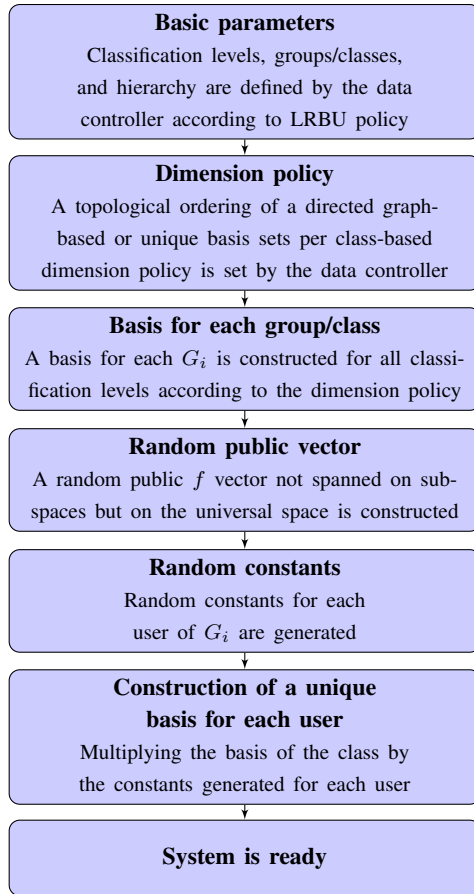


FIGURE 4. Steps of system-ready phase

parent which is not feasible to guess from its own basis. For example, if W_1 is a three dimensional subspace of \mathbb{R}^n ($n \geq 4$) and $W_1 \supset W_2$ of dimension two, it is not feasible to obtain W_1 from W_2 . In fact, there are uncountable subspaces of \mathbb{R}^n of three-dimension, and if only two basis elements are known, it is impossible to obtain the third one. The general framework of extracting a K_{G_i} from a known S_i for the W is described in Figure 3, and it is the point on which the security of our scheme are based. On the other hand, users belonging to the same classification level but different classes (e.g. G_2 and G_3), he/she can access the key only if he/she knows all the basis elements of the class he/she wants to access. This process will be determined according to the selected dimension policy (a topological ordering of a directed graph-based or a unique basis sets per class) as it will be explained below.

D. DYNAMIC UPDATE PHASE

The hierarchical key assignment/access control schemes should support the change in the topology over time. For this reason, distinct dimension policies can be determined in advance by taking into account that the number of classes and classification levels in the hierarchy can change.

Firstly, our proposed scheme allows the dynamic updates

(class insertion/deletion) in the structure without needing to construct new public information f used by any G_j for for all the following policies. On the other hand, the need for private information and time-cost variations will arise according to the six distinct dimension policies set by the data controller as follows:

- 1^{th} dimension policy. A topological ordering of a directed graph-based inflexible/unscalable dimension policy.
- 2^{nd} dimension policy. A topological ordering of a directed graph-based flexible/scalable dimension policy.
- 3^{rd} dimension policy. A unique basis sets per class-based inflexible/unscalable dimension policy.
- 4^{th} dimension policy. A unique basis sets per class-based flexible/scalable dimension policy - a new class tolerable.
- 5^{th} dimension policy. A unique basis sets per class-based flexible/scalable dimension policy - new classification level tolerable.
- 6^{th} dimension policy. A merge of 4^{th} and 5^{th} dimension policies.

Insertion. Let's take into consideration that a class at the same classification level can have the same basis set (each class does not have its own unique basis set), but this policy will cause the hierarchy to be like a topological ordering of a directed graph (1^{th} dimension policy). In this case, siblings with the same ancestors at the same classification level have the basis set for the same subspace and therefore the secret key would be the same (see Figure 6 and Table 1). If a new class G_j is inserted into the structure as a member of one of the current classification levels, there is no need to update any basis sets.

TABLE 1. The data controller's 1^{th} dimension policy for a topological ordering of a directed graph - inflexible/unscalable. * Each tuple in the third column represents the dimension of basis of each class/group in its classification level.

Class	classification Level	Number of dimension
G_1	Top Secret	$b + 2$
G_2	Secret	$b + 1$
G_3	Confidential	b

On the other hand if a new class G_j is inserted into the hierarchical structure as a new member out of the current classification levels, the inflexible/unscalable dimension policy does not work, in which case the data controller needs to apply the 2^{nd} dimension policy (See Table 2).

In the 2^{nd} dimension policy, the data controller can flexibly design the hierarchy and define the number of dimensions of each classification level so that there is no need to update the basis sets when new classes G_j are inserted into the hierarchy.

As it can be seen in Table 2, under each classification level, a maximum of 10 new classification levels can be defined quickly and easily without any update in the hierarchy. The number of classes G_i at each classification level does not matter as members will use the same basis set and secret key.

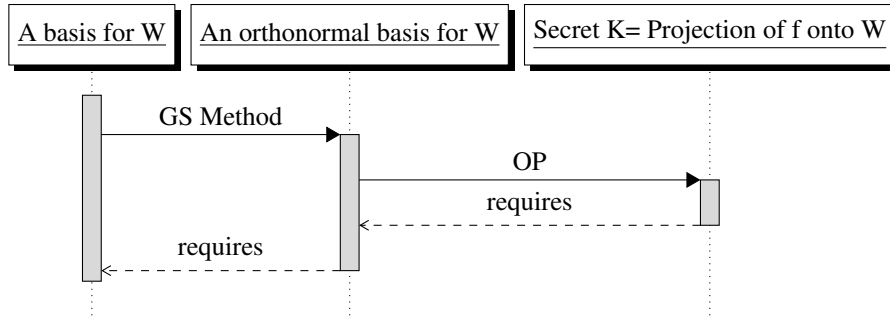
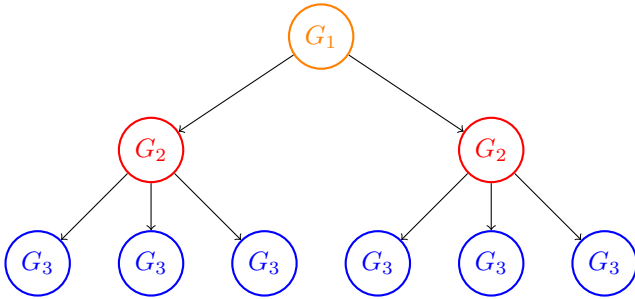


FIGURE 5. Steps of the key derivation phase.

FIGURE 6. The hierarchy of same basis sets for same classification levels.
*Orange: top secret, red: secret, blue: confidentialTABLE 2. The data controller's 2^{nd} dimension policy for a topological ordering of a directed graph - flexible/scalable * Each tuple in the third column represents the dimension of basis of each class/group in its classification level.

Class	classification Level	Number of dimension
G_1	Top Secret	$b + 22$
G_2	Secret	$b + 11$
G_3	Confidential	b

In the inflexible 3^{rd} dimension policy (Table 3), assume that a hierarchy in which each class has its own unique basis set (See Figure 1, and Section IV) and each class's number of children is two, that of the leaf classes' is zero. This policy can be adopted for inflexible/unscalable and change-resistant hierarchies that do not allow inserting a new class, and classification levels, as well as the initial topology, needs to remain constant.

TABLE 3. The data controller's inflexible/unscalable 3^{rd} dimension policy. * Each tuple in the third column represents the dimension of basis of each class/group in its classification level.

Class	classification Level	Number of dimension
G_1	Top Secret	$4b + 3$
G_2, G_3	Secret	$2b + 1$
G_4-G_9	Confidential	b

In the flexible 4^{th} dimension policy, which is the extension of the 3^{rd} dimension policy, the data controller considers that the number of children in each class will increase by 1 over time like in Figure 7, and the initial topology can be created as in Table 4. Each class's number of children is three, and that of the leaf classes' is zero. This policy

can be adopted for flexible/scalable hierarchies that do allow inserting new classes into the hierarchy without any change in the classification levels. The data controller will only apply steps 4 and 5 of the system-ready phase for users of newly added classes.

TABLE 4. The data controller's flexible/scalable 4^{th} dimension policy. * Each tuple in the third column represents the dimension of basis of each class/group in its classification level.

Class	classification Level	Number of dimension
G_1	Top Secret	$9b + 4$
G_2, G_3, G_8	Secret	$3b + 1$
G_4-G_7, G_9-G_{13}	Confidential	b

If a new class G_j is inserted into the hierarchy as a member out of the current classification levels, a new basis set S_j needs to be given to this new class as private information concerning its classification level by the data controller. Then, all related parents with G_j will have the updated basis set, including the basis elements of the S_j . In this case, there is no need to update any child's basis set. But, the data controller can prefer the 5^{th} dimension policy so that the relevant parents do not need to update their basis sets.

In the flexible 5^{th} dimension policy, which is another extension of the 3^{rd} dimension policy, the data controller considers that the number of classification level will increase by 1 over time like in Figure 8, and the initial topology can be created as in Table 5. Each class's number of children is three, and that of the leaf classes' is zero. This policy can be adopted for flexible/scalable hierarchies that do allow inserting new classification level into the hierarchy. The data controller will only apply steps 4 and 5 of the system-ready phase for users of newly added classes.

TABLE 5. The data controller's flexible/scalable 5^{th} dimension policy. * Each tuple in the third column represents the dimension of basis of each class/group in its classification level

Class	classification Level	Number of dimension
G_1	Top Secret	$27b + 13$
G_2, G_3, G_8	Secret	$9b + 4$
G_4-G_7, G_9-G_{13}	Confidential	$3b + 1$
$G_{14}-G_{40}$	Unclassified	b

Apart from the above-mentioned dimension policies, the data controller can construct a new dimension policy that

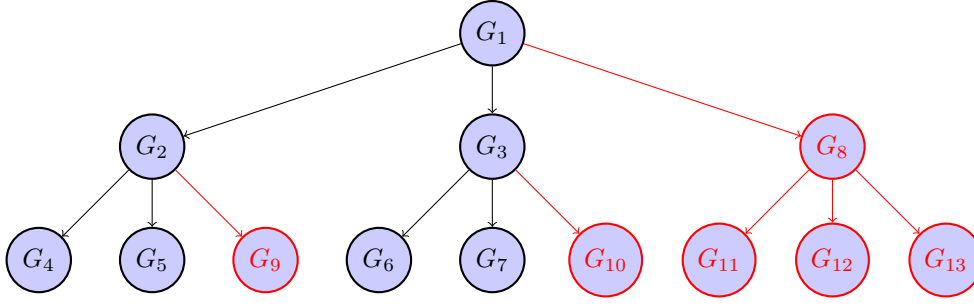


FIGURE 7. Updated poset sample (red-colored classes are inserted) in hierarchy. *Updated from perfect binary tree of depth 3 with sequential class/group numbers, a full binary tree of depth k having $2^k - 1$ nodes, $k \geq 0$.

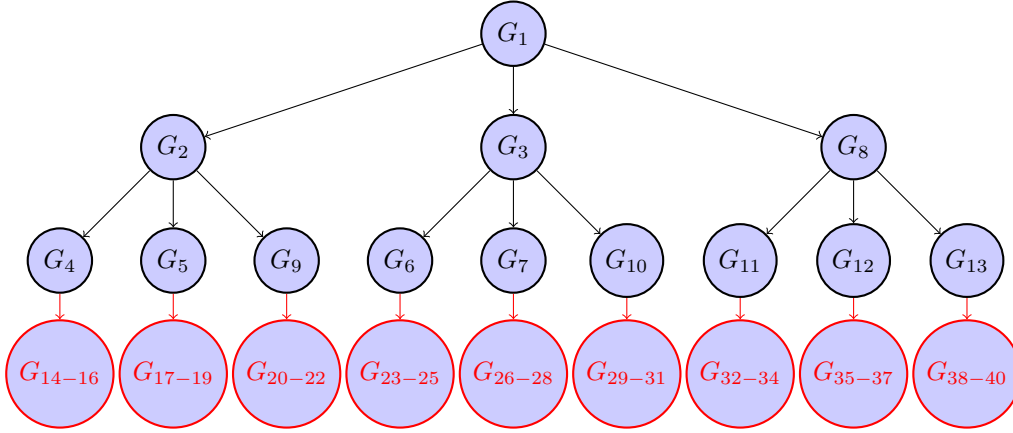


FIGURE 8. Updated poset sample (red-colored classes are inserted as a member of unclassified classification level) in hierarchy. *Updated from Figure 7.

uses a merge of the 4th and 5th dimension policies without applying steps 1-3 of the system-ready phase to create a flexible topology suitable for expansion in both the horizontal and vertical directions.

Deletion: As can be seen in Figure 9, if any G_j which is a parent is deleted from the hierarchy, then all children of the deleted parent will be linked to one higher parent class. However, if the deleted class is at the bottom of the hierarchy, there will be no change in the linkage. In addition, if the data controller deletes any G_j in the hierarchy, the basis set of G_j will also be deleted from the higher parent classes to ensure efficiency and reduce storage need.

V. EXPERIMENTAL RESULTS

We implemented the proposed hierarchical key access scheme on a computer with Macintosh OS Monterey running on 3.2 GHZ 6-Core Intel Core i7 processor, 16 GB 2667 Mhz DDR4 memory, and we use SageMath and Jupyter Notebook.

Algorithm 1 stands for both the system-ready and key derivation phases. The derivation process of a secret key K_i requires first finding an orthonormal basis described in Algorithm 2 (GS orthogonalization process). Upon constructing an orthonormal basis, the method employs OP, which outputs the K_i . The vector space V can be selected as an inner product space, and in fact, V being a polynomial space

over any field offers suitable choices for inner product and subspaces. On the other hand, for simplicity, we employed the well-known space \mathbb{R}^n and the inner product, which is called dot product, in the implementations.

A. PERFORMANCE ANALYSIS

In this section, we will discuss the performance analysis of the proposed scheme.

Public and private storage needs, system-ready, key derivation, and key update overheads, and change in the hierarchy are critical metrics to gauge the efficiency of our scheme. Note that c denotes the number of classes G in the hierarchy, b denotes the number of dimensions of a subspace associated with any class at the bottom of the hierarchy.

Public information storage need. The f vector which does not lie in the subspace W_i for all $i = 1, 2, \dots, c$ where c is the total number of classes in the hierarchy but lies in the vector space V is the only public information for each class to derive the corresponding secret keys.

Private information storage need. Each member of the class needs its own basis S_{U_i} to derive the secret keys. The S_{G_i} , which belongs to the bottom classes, is the private information of the corresponding class. Therefore, the maximum private storage need per class in the hierarchy is b . All elements of the basis sets of the children are also the member

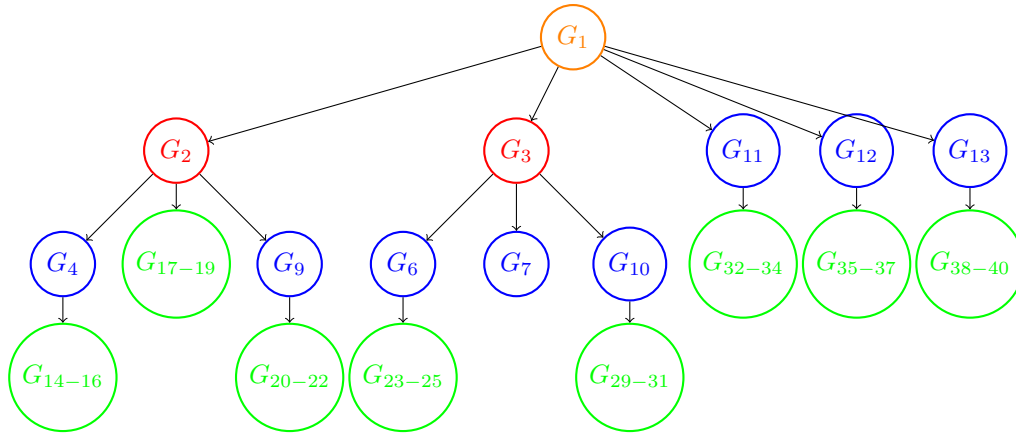


FIGURE 9. G_8 (secret classification level), G_5 (confidential classification level), and G_{26}, G_{27}, G_{28} (unclassified classification level) are deleted *Updated from Figure 8.

Algorithm 1:

Input:

- 1 A subspace W and its basis $\{b_1, b_2, \dots, b_n\}$ where $W \subset V$ infinite dimension
- 2 $U_{ij}, i = 1, 2, \dots, c, j = 1, 2, \dots, z$ where c is the number of classes and z is the number of users in c^{th} class
- 3 $k_i, i = 1, 2, \dots, dim$ where dim is the dimension

Initialization:

- 4 $f \leftarrow \text{selectRandomVector}(V)$ where $f \in V$
- 5 $S_0 \leftarrow W$ where $f \notin W$ and $W \subset V$

Forward Inclusion:

- 6 **for** $i = 1$ **to** c **do**
 - 7 $S_i \leftarrow \text{span}\{S_{i-1}^{k_i}\}$ select the first k_i vectors in S_{i-1} where $S_{i-1} \supset S_i$
 - 8 **for** $j = 1$ **to** z **do**
 - 9 $C_{ij} \leftarrow \text{RandomIntVector}(L)$ where $L \in R^{k_i}$
 - 10 $U_{ij} \leftarrow S_i \cdot C_{ij}$
· represents component-wise multiplication
 - 11 $W_{ij} \leftarrow \text{gramSchmidt}(U_{ij})$ see Algorithm 2
 - 12 $K_{ij} \leftarrow \sum_{i=1}^n \langle f, w_i \rangle \cdot w_i$
-

of relevant parent classes' basis sets. Thus, all parents have one additional private vector, which is not on the basis of any of its children. In summary, all classes except the lowest child classes at the bottom have only one private information, whereas the classes at the bottom have a maximum private information need of b . However, if the data controller adopts a flexible/scalable dimension policy for the hierarchy, namely taking the scalability into consideration, private storage need can increase, thus there is a trade-off between private storage

Algorithm 2:

Input:

A Vector set $V = \{v_1, v_2, \dots, v_n\}$

Procudure gramSchmidt(V):

- 1 $w_1 = \frac{v_1}{\|v_1\|}$
- 2 **for** $j = 2$ **to** n **do**
 - 3 **for** $k = 1$ **to** $j-1$ **do**
 - 4 $w_j \leftarrow v_j - \frac{\langle v_j, w_k \rangle}{\langle w_k, w_k \rangle} w_k$
 - 5 $w_j = \frac{w_j}{\|w_j\|}$

Return:

Orthonormal vector set $V' = \{w_1, w_2, \dots, w_n\}$

need and scalability.

System-ready and key derivation time-cost.

The secret key K of a class is derived from users' private information and the public information f and allows accessing to the data. The key derivation process involves GS orthogonalization operation and projection of the public vector f on the subspace generated by the users' private basis elements. The actual complexity depends on the size of the basis and the defined inner product on the universal set V .

To compute the complexity of these steps, we fixed the class G and the corresponding subspace, which W denotes. Let n be the dimension of the subspace W , which is assigned to the class G . We will present the key derivation complexity in three phases: the number of inner product I , multiplication M , and division D operations.

The number of I : $n^2 + n$.

1) GS orthogonalization process:

$$\sum_{i=1}^n 2(i-1) = n^2 - n \text{ inner products are required.}$$

- 2) Normalization process: The orthogonal basis should be normalized, requiring n additional I for n vectors.
- 3) The final operation for the projection is the determination of the coefficients in the equation 2 which requires n more I for the n dimensional vector space.

The number of M : $\frac{(n^2 + n)}{2}$.

- 1) GS orthogonalization process:

$$\sum_{i=1}^n (i-1) = \frac{n(n-1)}{2} \text{ multiplications.}$$

- 2) Equation 1: n additional M must be performed to obtain the projection vector h .

The number of D : $\frac{(n^2 + n)}{2}$.

- 1) GS orthogonalization process:

$$\sum_{i=1}^n (i-1) = \frac{n(n-1)}{2} \text{ divisions.}$$

- 2) Normalization process: n additional D is required to normalize the orthogonal set consisting of n vectors.

In summary, the secret key K derivation complexity for a user is $O(n^2)$ where the ultimate complexity is

$$(n^2 + n)I + \frac{n^2 + n}{2}M + \frac{n^2 + n}{2}D.$$

The system-ready time-cost means that the system is ready for the key derivation process for users in the organization and includes the steps from 1 to 9 in Algorithm 1. Whereas, to calculate the key derivation time-cost for any user U in the class G , only for the 11th and 12th steps of Algorithm 2 have been considered.

We implemented Algorithm 1 and Algorithm 2 while selecting real vector space \mathbb{R}^n over real numbers to observe the real-time-costs. These include both the system-ready and key derivation phases. The inner product is chosen to be the dot product.

For simplicity of experimental results, a full binary tree with sequential class/group numbers (see Figure 10) and full non-binary tree hierarchical structure where the number of children is three are utilized. Note that b is chosen to be 3, and the dimensional policy determined by the data controller is taken into account.

Experimental results are shown in Figure 11 – Figure 15. These results satisfy all of the desired performance requirements. The results indicate that our proposed scheme efficiently and flexibly applies all dimension policies defined by the data controller in a dynamic hierarchical environment.

The system-ready time-cost is tolerable and within acceptable limits and the key derivation time-cost is linear with the number of classes in the hierarchical structure. For example, even in an organizational structure with 364 groups/classes

and with 850 top secret classes, the system's ready time-cost is approximately 1.5 seconds (see Figure 15). In addition, in organization with 40 groups/classes where the number of dimension of top secret class is 94, the system's ready time-cost is approximately 22 milliseconds (see Figure 14).

On the other hand, the key derivation time-cost is linear with the number of classes in the hierarchical structure. For example, in a hierarchy with 255 groups/classes and the number of dimension of the highest classification level is 10, the key derivation time-cost is approximately 14 milliseconds (see Figure 12), or in a hierarchy with 40 groups/classes and the number of dimension of top secret class is 94, the key derivation time-cost is approximately 743 milliseconds (see Figure 14). Note that key derivation time-cost is actually the time-cost of calculating an OP.

Key update. Note that each class in the hierarchy has its own secret key. If it is compromised or needs to be changed, then private information (basis sets) would be redistributed again, thus the system-ready process would run again and the system-ready time-costs will only be incurred. On the other hand, the public information and key derivation time-cost would remain the same.

B. SECURITY ANALYSIS

In this section, we will discuss the security analysis of the proposed scheme only if one of the unique basis sets per class-based dimension policies is chosen by the data controller.

Key recovery secure. Because each class in the hierarchy has a distinct basis S , any child class, and any same level class cannot derive any secret keys that do not belong to their own class. Our scheme is collusion resistant to coalition attacks of the child classes and the same level classes. Hence we can easily say that it is KR_s .

If some child classes collude and try to get the key of their parent class, then they have to find a unique linearly independent vector of its parent class, which is computationally infeasible. The argument is also valid for the same level classes. In other words, the members of a class can not find the basis set of the same classification level class.

Key indistinguishability secure. According to this principle, the attacker should not distinguish between the secret key and a random string of the same length. Our scheme is secure under the assertion of chosen linearly independence vectors. For any child class, finding a unique linearly independent vector of the parent class is negligible. In other words, any element in the universal not belonging to the subspace associated with the child is equally likely to be the one that the child class is looking for. That makes finding an upper-class basis computationally infeasible. Thus, our scheme is KI_s .

Resistance to privilege creep problem. If there is a change in the hierarchy, which means that the classification level of any U of the class is downgraded or the membership is changed (e.g., if a U member of G_7 later becomes a member of G_8 , or leaves the organization), the user U should

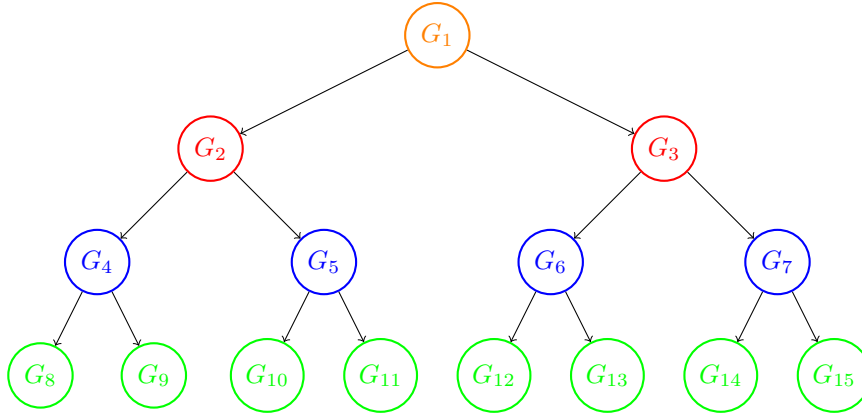


FIGURE 10. Perfect binary tree: Full binary tree of depth 4 with sequential class/group numbers. A full binary tree of depth k having $2^k - 1$ nodes, $k \geq 0$. *Orange: top secret, red: secret, blue: confidential, green: unclassified.

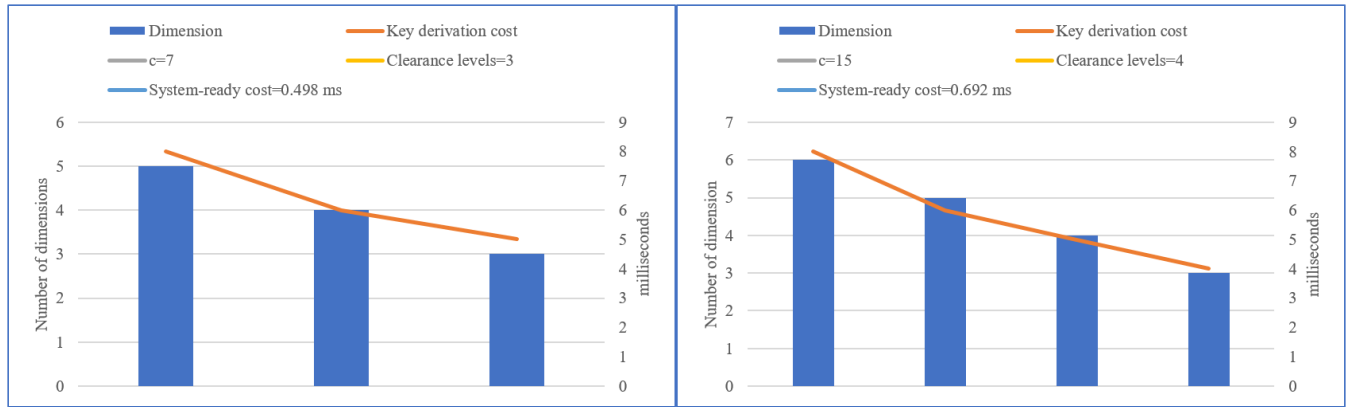


FIGURE 11. 1st dimension policy: A topological ordering of a directed graph-based inflexible/unscalable dimension policy. * Experimental results of a full-binary tree hierarchy with three and four different classification/clearance levels.

no longer be able to access other classes for which they previously had privileges. When any of the above situations occur, the basis S owned by the user's class gets extra basis element v_i distributed to all users in this class and relevant parents (inserting only one element into the basis of the relevant classes). In this way, the system-ready process will run again, the maximum private storage will increase by the same as relevant space's dimension increases by one, and the key derivation time-cost will increase accordingly. The left user can not guess the extra basis elements, so the user can not derive the key anymore at the expense of a little extra time-cost.

C. COMPARISON WITH OTHER SCHEMES

Table 6 reveals a comparison between ours and some other well-known key access schemes. The comparison takes into account several parameters, and the notation can be seen at the bottom of Table 6.

In Akl and Taylor based schemes (see "Literature Review") [2], [6] – [13], [16], [17], the key update and the key derivation are costly that these schemes might not be practical for large distinct classes in a hierarchical structure. Some are insecure against collusion attacks and only KR_s and KI_s

under the RSA assertion or random exponent. Note that both KR_s and KI_s , in other words, a provably secure scheme first came up with [15].

In [15] and [30], the number of public information increases with the number of security classes. Private storage is only a single symmetric key for each class. However, the key derivation time-cost is directly related to the path length between classes. Namely, the time-cost grows if the path length between classes increases. They are KI_s and chosen-plaintext-attack-resistant CPA_r according to the security perspective.

In [14], it has proven that Akl and Taylor-based schemes [6] – [13], [16], [17] (KR_s by default) can only be KI_s under RSA assertion RSA_s . But, to be KI_s , the additional public storage need and key derivation time-cost must be met, which will be a quite multiple of the bit length l of the key.

In [18], the key derivation is very efficient even if the path length between classes is too much. The main drawback are storage needs. For the first scheme, both the number of time periods and classes are critical for determining the public storage need, on the other hand, for the second, only the number of time periods is a key parameter to determine the private storage need.

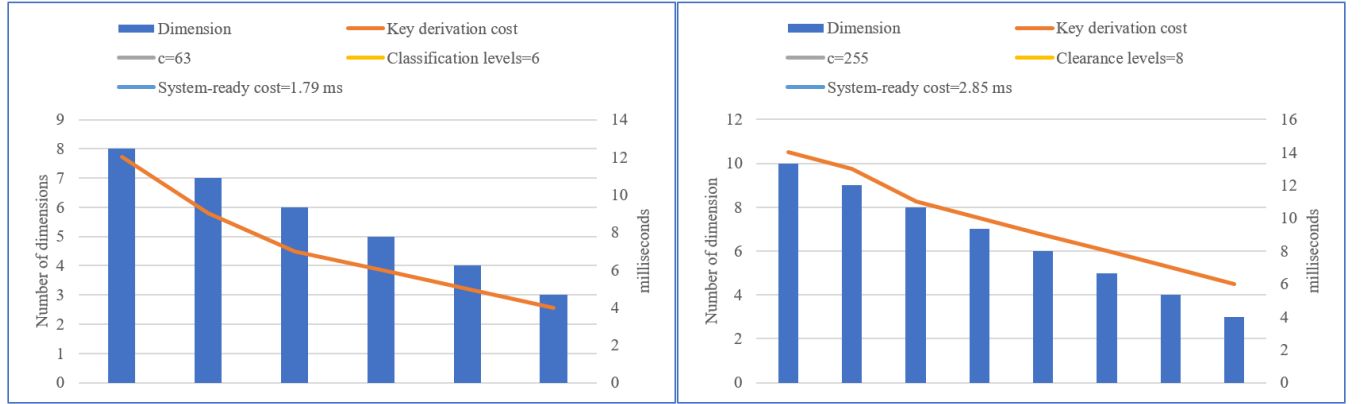


FIGURE 12. 2nd dimension policy: A topological ordering of a directed graph-based flexible/scalable dimension policy. * Under each classification level, one more classification level can be defined by the data controller. * Experimental results of a full-binary tree hierarchy with six and eight different classification/clearance levels.

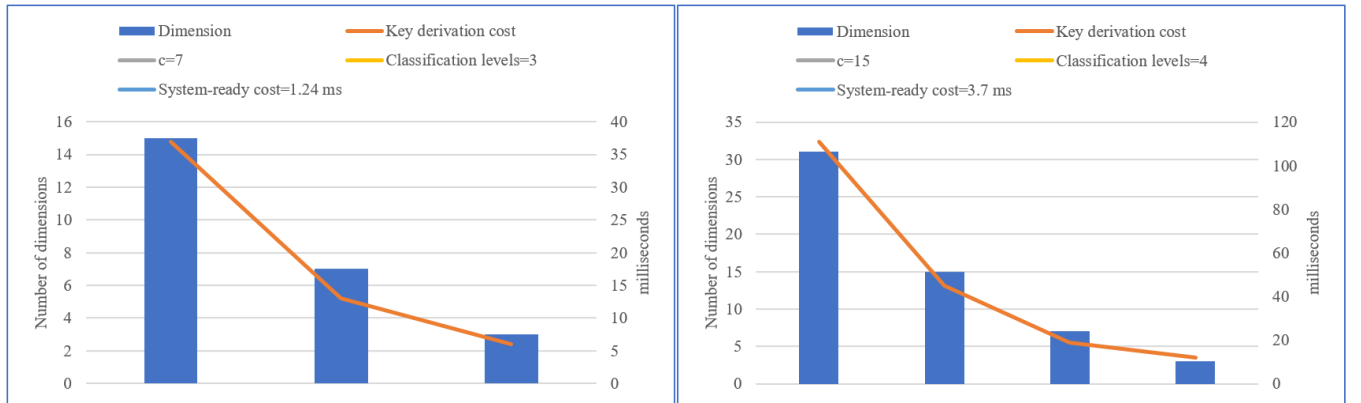


FIGURE 13. 3th dimension policy: A unique basis sets per class-based inflexible/unscalable dimension policy. * Experimental results of a full-binary tree hierarchy with three and four different classification/clearance levels.

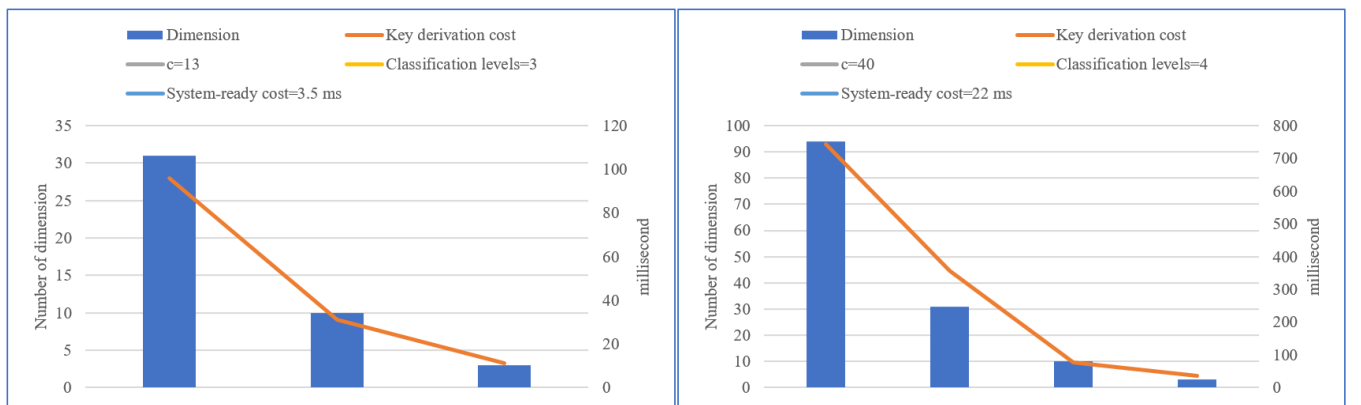


FIGURE 14. 4th and 5th dimension policy: A unique basis sets per class-based flexible/scalable dimension policy. * Experimental results of a full non-binary tree hierarchy, the number of children is three, with three and four different classification/clearance levels.

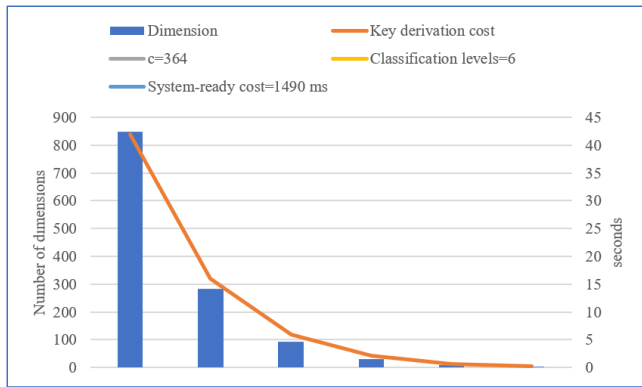


FIGURE 15. 6th dimension policy: A merge of (both fourth and fifth) dimension policies. * Experimental results of a full non-binary tree hierarchy, the number of children is three, with six different classification/clearance levels.

In [23], there is a trade-off between private information storage need and efficiency of key derivation. The ultimate efficiency of key derivation is restricted to the path length between classes in hierarchy h . On the other hand, the amount of private information relies on the poset width w . The most important contribution to the literature is that it does not need public storage.

In [32], the direct access scheme without the need for iterative computation is SKI_s and addresses the potential hierarchical changes with minimum computations. But the main drawback is public storage need in comparison with others, but there is a linear relation between storage need and computational cost. Each class in the hierarchy requires to calculate two times M (modular multiplication cost), and one time A (modular addition cost) over F_q for the derivation of its K . In addition, each class requires to calculate twice the value of F , four times M , and two times A over F_q . Thus, the ultimate cost of each class might be intolerable, which makes the scheme inefficient. Note that the change/update in hierarchy requires the data controller to calculate and publish a novel public matrix.

In [34], the private storage need is equal to the number of classes G in specific OU , but it can be reduced to 1, on the other hand, the public storage need is equal to zero. In addition, the key derivation time-cost is equal to the square of the interpolation polynomial degree d^2 which means the time-cost is quadratic.

On the other hand, our proposed scheme provides fine-grained and flexible hierarchical direct key assignment/access control and does not need other classes for the key derivation. Although, the key derivation complexity is n^2 , where n stands for the dimension of the corresponding vector space, the experimental results satisfy all desirable performance requirements. The results indicate that our proposed scheme efficiently applies all dimension policies in a dynamic hierarchy. The system-ready time-cost is within acceptable limits and the key derivation time-cost is linear with the number of classes in the hierarchical structure. Our scheme's private and public storage needs are tolerable and

it is SKI_s based on the security of the linearly independent chosen vectors $LICV$. The other advantage over [32] is that there is no need to publish new public information if there is a change in the hierarchy.

VI. CONCLUSION

In this work, the inner product spaces are applied to construct a hierarchical key assignment scheme employed in various situations, particularly for any cloud service.

Private and public storage need, key derivation time-cost, and key update, resistance to collusion attacks and privilege creep problem, the state of KR_s and KI_s , and whether creating an information-theoretic secure system or not are critical parameters to compare our schemes with other hierarchical assignment/access control schemes in the literature. For instance, private and public storage needs are among the key burdens for the data controller who processes the mission-critical data in hierarchical infrastructures. Our scheme has reduced these to certain low levels.

The other best practice for key assignment schemes ensuring a computationally efficient method for key derivation and updating is also provided in our scheme.

According to the security perspective, the collusion problem in other words collaborated attack, which is one of the main problems for key access assignment schemes in literature is prevented by our scheme. Our scheme guarantees the resistance to collusion attacks and privilege creep problem, ensuring forward and backward security, so it is both KR_s and KI_s .

REFERENCES

- [1] A. Shamir, "How to Share a Secret", Communications of ACM, 22(11), 612-613, 1979.
- [2] S.G. Akl and P.D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy", ACM Transactions on Computer Systems, vol. 1, no. 3, pp. 239-248, 1983.
- [3] S. Kamara and K. Lauter, "Cryptographic Cloud Storage", Financial Cryptography and Data Security, 136-149, 2010.
- [4] X. Dong, J. Yu, Y. Luo, Y. Chen, G. Xue, and M. Li, "Achieving Secure and Efficient Data Collaboration in Cloud Computing", IEEE/ACM 21st International Symposium on Quality of Service, Montreal, QC, pp. 1-6, 2013.
- [5] L. Zhou, V. Varadarajan, and M. Hitchens, "Achieving Secure Role-Based Access Control on Encrypted Data in Cloud Storage", IEEE Transactions on Information Forensics and Security, vol. 8, no. 12, pp. 1947-1960, 2013.
- [6] S. MacKinnon, P. Taylor, H. Meijer, and S. Akl, "An Optimal Algorithm for Assigning Cryptographic Keys to Control Access in a Hierarchy" IEEE Transactions on Computers vol. 34, no. 9, pp. 797-802, 1985.
- [7] R. Sandhu, "Cryptographic Implementation of a tree hierarchy for access control", Information Processing Letters, vol. 27, No. 2, pp. 95-98, 1988.
- [8] L. Ham, H.Y. Lin, "A cryptographic key generation scheme for multilevel data security", Computers and Security, vol. 9, No. 6, pp. 539-546, 1990.
- [9] C.C. Chang, R.J. Hwang and T.C. Wu, "Cryptographic key assignment scheme for access control in a hierarchy", Information Systems, 17 (3), 243-247, 1992.
- [10] H.T. Liaw, S.J. Wang and C.L. Lei, "A dynamic cryptographic key assignment scheme in a tree structure", Computers Math. Applic. 25 (6), 109-114, 1993.
- [11] M.S. Hwang, C.C. Chang and W.P. Yang, "Modified Chang-Hwang-Wu access control scheme", IEEE Electronics Letters 29 (24), 2095-2096, 1993.
- [12] H.T. Liaw and C.L. Lei, "An optimal algorithm to assign cryptographic keys in a tree structure for access control", BIT 33, 46-56, 1993.

TABLE 6. Comparison with Previous Schemes

Schemes	Dynamic	Private Storage	Public Storage	Key Derivation	Security Type	Security Assertions
Akl and Taylor based [2] [6] – [13], [16], [17]	No	One	$c.t$	t_{exp}	KR_s	N/A
Atallah et al. [15]	Yes	Three (max)	$e.t$ (min)	$h.(DT_{se}+T_{prf})$	KI_s	CPA_s+PRF
De Santis et al. TBEBF [30]	Yes	One (min)	$e.t$ (min)	$h.DT_{se}$ (min)	KI_s	CPA_s
De Santis et al. TBEBF [30]	Yes	One (min)	$c.t$ (min)	DT_{se} (complex)	KI_s	CPA_s
D'Arco et al. [14]	Yes	One	$l.c$ (min)	$l.DT_{se}$	KI_s	RSA_s
Ateniese et al. TLEBC [18]	Yes	One	$c^2.t^3$ (max)	DT_{se}	KI_s	CPA_s
Ateniese et al. TLPBC [18]	Yes	t (max)	c^2 (max)	DT_p	KI_s	CPA_s
Freire et al. PRF-based [23]	No	w	Zero	$h.T_{prf}$	SKI_s	PRF
Freire et al. FSPRG-based [23]	No	w	Zero	$h.T_{prg}$	SKI_s	$FSPRG$
Tang et al. [32]	Yes	Four	$c^2 + 1$ (min)	$2A+4M$	SKI_s	PRF
Celiktas et al. [34]	Yes	One (min) c (max)	Zero	$d^2 \cdot \log p$	KI_s	$TSBITS$
The proposed scheme	Yes	One (min) b (max)	One	OP_t	SKI_s	$LICV$

Notation:

c : number of classes
 t : number of time period
 t_{exp} : exponential time over a huge group/class ($D+E$).
 KR_s : key recovery secure
 N/A : Not applicable
 e : number of edges
 h : path length between the classes
 DT_{se} : decryption time using a symmetric key encryption scheme
 T_{prf} : time of PRF evaluation
 KI_s : key indistinguishability secure
 CPA_s : chosen-plaintext-attack-resistant
 PRF : pseudorandom function
 l : bit length of the key
 RSA_s : secure under the RSA assertion
 DT_p : decryption time using pairing evaluation
 w : poset width
 SKI_s : strong key indistinguishability secure
 T_{prg} : time of 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 : interpolation polynomial degree
 p : the number of elements in F_p
 $TSBITS$: threshold scheme-based information-theoretic secure
 b : maximum dimension of a subspace associated with any class at the bottom in the hierarchy
 OP_t : time-cost using orthogonal projection
 $LICV$: linearly independent chosen vectors

- [13] Hwang Min-Shiang, "A cryptographic key assignment scheme in a hierarchy for access control", Mathematical and Computer Modelling, vol. 26, No. 2, pp. 27-31, 1997.
- [14] P. D'Arco, A. De Santis, A. L. Ferrara, B. Masucci, "Variations on a theme by Akl and Taylor: Security and tradeoffs", Theoretical Computer Science, vol. 411, no.1, pp. 213-227, 2010.
- [15] M. Atallah, M. Blanton, N. Fazio, and K. Frikken, "Dynamic and Efficient Key Management for Access Hierarchies", ACM Transactions on Information and System Security, vol. 12, No. 3, Article 18, 2009.
- [16] W-G. Tzeng, "A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy", IEEE Transactions on Knowledge and Data Engineering, Vol.14, No.1, January/February 2002.
- [17] H. Chien, "Efficient Time-Bound Hierarchical Key Assignment Scheme", IEEE Transactions on Knowledge and Data Engineering, 16, 1301-1304, 10.1109/TKDE.2004.59, 2004
- [18] G. Ateniese, A. De Santis, A. L. Ferrara, and B. Masucci, "Provably-Secure Time-Bound Hierarchical Key Assignment Schemes", Journal of Cryptology, vol. 25, no.2, pp. 243-270, 10.1007/s00145-010-9094-6, 2012.
- [19] X. Yi, "Security of Chien's efficient time-bound hierarchical key assignment scheme," IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 9, pp. 1298-1299, Sept. 2005, doi: 10.1109/TKDE.2005.152
- [20] Xun Yi and Yiming Ye, "Security of Tzeng's time-bound key assignment scheme for access control in a hierarchy," IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 4, pp. 1054-1055, July-Aug. 2003, doi: 10.1109/TKDE.2003.1209023.
- [21] C. Jason, M. Keith, and W. Peter, "On Key Assignment for Hierarchical Access Control", Proceedings of the Computer Security Foundations Workshop 2006, 98-111. 10.1109/CSFW.2006.20, 2006.
- [22] V. R. Shen and T.-S. Chen, "A novel key management scheme based on discrete logarithms and polynomial interpolations," Comput. Security, vol. 21, no. 2, pp. 164-171, 2002
- [23] E.S.V. Freire, K.G. Paterson, and B. Poettering, "Simple, Efficient and Strongly KI-Secure Hierarchical Key Assignment Schemes", In: Dawson E. (eds) Topics in Cryptology – CT-RSA 2013. CT-RSA 2013. Lecture Notes in Computer Science, vol 7779. Springer, Berlin, Heidelberg.
- [24] E. Bertino, N. Shang and S. S. Wagstaff Jr., "An Efficient Time-Bound Hierarchical Key Management Scheme for Secure Broadcasting," IEEE Transactions on Dependable and Secure Computing, vol. 5, no. 2, pp. 65-70, April-June 2008, doi: 10.1109/TDSC.2007.70241.
- [25] Y. F. Chung, H.H. Lee, F. Lai, and T. S. Chen, "Access control in user hierarchy based on elliptic curve cryptosystem", Information Sciences, vol. 178, no. 1, pp. 230-243, 2008.
- [26] F.G. Jeng, C.M. Wang, "An efficient key-management scheme for hierarchical access control based on elliptic curve cryptosystem", Journal of Systems and Software, vol. 79, no. 8, pp. 1161-1167, 2006.
- [27] H.M. Sun, K.H. Wang, and C.M. Chen, "On the security of an efficient time-bound hierarchical key management scheme," IEEE Transactions on

- Dependable and Secure Computing, vol. 6, no. 2, pp. 159–160, 2009.
- [28] A.K. Das, N. R. Paul, L. Tripathy, "Cryptanalysis and improvement of an access control in user hierarchy based on elliptic curve cryptosystem", *Information Sciences*, vol. 209, pp. 80–92, 2012.
- [29] Yu-Li Lin, Chien-Lung Hsu, "Secure key management scheme for dynamic hierarchical access control based on ECC", *Journal of Systems and Software*, vol. 84, no. 4, pp. 679–685, 2011.
- [30] A. D. Santis, A. L. Ferrara, and B. Masucci, "Efficient provably-secure hierarchical key assignment schemes," *Theoretical Comput. Sci.*, vol. 412, no. 41, pp. 5684–5699, 2011.
- [31] H. RagabHassen, H. Bettahar, A. Bouadballah, and Y. Challal, "An efficient key management scheme for content access control for linear hierarchies," *Comput. Netw.*, vol. 56, no. 8, pp. 2107–2118, 2012.
- [32] S. Tang, X. Li, X. Huang, Y. Xiang, and L. Xu, "Achieving Simple, Secure and Efficient Hierarchical Access Control in Cloud Computing", *IEEE Transactions on Computers*, vol. 65, no. 7, pp. 2325–2331, 2016.
- [33] B. Celiktas, I. Çelikbilek and E. Özdemir, "A Higher Level Security Protocol for Cloud Computing," 2019 4th International Conference on Computer Science and Engineering (UBMK), 2019, pp. 97–101, doi: 10.1109/UBMK.2019.8907019.
- [34] B. Celiktas, I. Celikbilek and E. Ozdemir, "A Higher-Level Security Scheme for Key Access on Cloud Computing," in *IEEE Access*, vol. 9, pp. 107347–107359, 2021, doi: 10.1109/ACCESS.2021.3101048.
- [35] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web", Technical Report, Stanford InfoLab, 1999.
- [36] M. Abadi et al. "Tensorflow: A system for Large-Scale Machine Learning", 12th USENIX symposium on operating systems design and implementation (OSDI 16), Savannah, GA, USA, pp. 265–283. 2016.
- [37] Guzey, Sueda and Kurt, Gunes Karabulut and Ozdemir, Enver, "A Group Key Establishment Scheme", *arXiv*, 2021, doi: 10.48550/ARXIV.2109.15037
- [38] 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. Security*, vol. 22, no. 5, pp. 453–456, 2003.
- [39] S. Zhong and T. Lin, "A comment on the Chen-Chung scheme for hierarchical access control," *Comput. Security*, vol. 22, no. 5, pp. 450–452, 2003.



ENVER OZDEMIR (Senior Member, IEEE) received BS in Mathematics from Middle East Technical University, Turkey in 2002 and Ph.D. in Mathematics, University of Maryland, College Park, the USA in 2009. He was a Research Fellow at CCRG, NTU, Singapore in 2010–2014. He is currently holding an Associate Professor position at Informatics Institute, Istanbul Technical University, and he is also the deputy director of the National Center for High-Performance Computing (UHeM). His research interests include Cryptography, Computational Number Theory, Network Security.



BARIS CELIKTAS received the BS in System Engineering from Turkish Military Academy, Turkey in 2008, MS in International Relations from Karadeniz Technical University in 2016, and MS in Applied Informatics from Istanbul Technical University in 2018. He is currently pursuing a Ph.D. degree in the Cyber Security Engineering and Cryptography Programme at the Institute of Informatics, Istanbul Technical University, and he is working as an IT System and Security Manager at NATO. His current research interests include Cyber Security, Network Security, Cloud Computing, and Cryptography.



SUEDA GUZEY received BS in Mathematics from Middle East Technical University, Turkey in 2017. She is currently a Ph.D. student at Cyber Security Engineering and Cryptography in the Institute of Informatics, Istanbul Technical University and she is also a research assistant at the same department. Her current research interests include Cryptography, Cyber Security, Number Theory, and Network Security.