

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

Baris Celiktas, Sueda Guzey, and Enver Ozdemir, *Member, IEEE*

Abstract—An inner product space-based hierarchical access control scheme is presented in this work. The proposed scheme can be utilized in any cloud delivery model where the data owner 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 owner, left-to-right and bottom-up (LRBU) policy can ensure much more flexibility and efficiency, especially during any change in the structure. For each class, the secret keys can be derived only when a predetermined subspace is available. Our scheme is resistant to collusion attacks and privilege creep problems, as well as providing key recovery and key indistinguishability security. The performance analysis also shows us that the data storage overhead is much more tolerable than other schemes in the literature. In addition, the other advantage of our key access scheme over many others in the literature is that it requires only one operation to derive the secret key of child classes securely and efficiently.

Index Terms—access control, hierarchical, inner product, key assignment, vector space.



1 INTRODUCTION

THE confidentiality of data in the digital medium is provided by employed 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 to that predetermined key is supposed to reflect the data owner's policy which might be complicated in certain institutions. The 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 clearance level determined by the data owner might not be 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, there are still open spaces to be completed towards a practical access mechanism for hierarchical structures which motivate us to conduct this research.

The recent trend of moving various services to a digital medium is welcomed by many institutions that process mission-critical data. Such institutions 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 owner'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 cost [5].

This work eliminates the hesitation to utilize the public cloud. It alleviates concerns about moving mission-critical data to the public cloud. A secure and flexible hierarchical inner product space-based key access scheme utilizing a mathematical tool of orthogonal projection on an inner product space is designed. 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 multilevel lattice-based model that addresses confidentiality in a hierarchical organizational structure similar to government or military institutions. 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 only 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 owner.

Various institutions 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 which we denote by *OU*. Some of the advantages of this mechanism are that user groups in each *OU* have different security classification levels to access the data, the members of lower classification level have to get approvals from the members of higher classification level, and the data owner can dynamically adjust all these. The scheme is presented to adapt such a hierarchical organizational structure to the digital environment. In addition, a cloud storage entity as a service provider should not be able to obtain any information about the user's mission-critical

• Authors are with Informatics Institute, Istanbul Technical University, Istanbul, Turkey, E-mails: celiktas16@itu.edu.tr, kayci19@itu.edu.tr, ozdemiren@itu.edu.tr

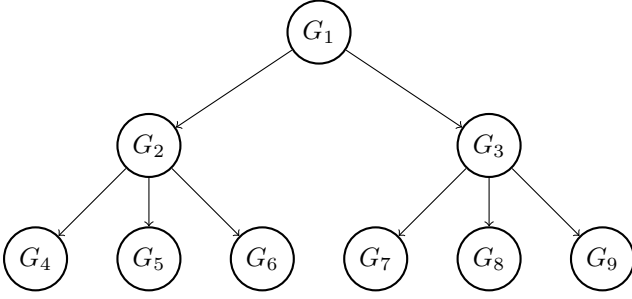


Fig. 1. Left to right and bottom to up policy-based partially ordered set (or poset) in hierarchy.

data. Thus, a scheme should facilitate the adoption of the public cloud and be employed safely for various purposes. The first stage of such a scheme that deals with the users' key access/assignment mechanism 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). A partition of set V is a collection of sets $\{V_1, V_2, \dots, V_n\}$. For example, the graph $G(9, 8)$ in Figure 1 is a DAG with nine vertices and eight directed edges. \leq is a partial order (transitive, reflexive, 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 $u, v \in V$ denote two distinct classes such that $u \leq v$. This relation means the users in class u can access all the data to which the users in class v have access. In other words, $G_i \leq G_j$ means that any user in G_i can access the data belongs to G_j , and the security clearance level of G_i is equal to or higher than that of G_j . Note that G_i has to be one of the parent classes of G_j . Figure 1 and Figure 2 illustrate the key access structure via the proposed algorithm.

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 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 only by the parent classes in addition to its own class, but the other way around is not allowed. Although they have the same clearance level (as shown in Figure 2), if they are not in the parent-child relationship in Figure 1, the key cannot be obtained since the relationship conditions are not met.

The remaining paper is structured as follows; Section 2 provides related works regarding hierarchical key assignment schemes in literature chronologically. Section 3 presents the preliminaries on which our work is based, such as the inner product space, the Gram-Schmidt method, and the orthogonal projection. Section 4 is devoted to the detailed presentation of the proposed scheme. Section 5 shows the implementation results, security and performance analysis of the proposed fully hierarchical key assignment scheme. Section 6 provides a summary of the proposed scheme.

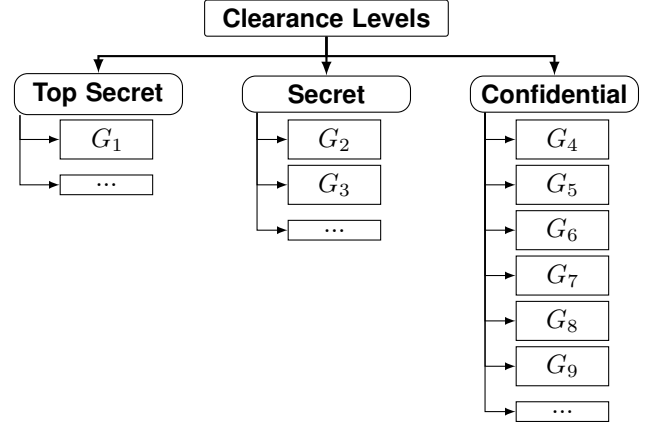


Fig. 2. Taxonomic ranks of clearance levels

2 RELATED WORK

This section will present the studies related to the hierarchical access control scheme in the literature chronologically.

To provide a key assignment scheme obeying a hierarchical structure, various schemes have been presented, and almost all hierarchical access control schemes [2], [6] – [13] are based on a partially ordered set (poset) hierarchy. These schemes are not designed to give access to users for a certain period of time only. The other ignored point in these schemes is that updating keys are based on poset hierarchy. In other words, the key derivation might be so costly that these schemes become non-practical for large hierarchies. We should note here that Akl and Taylor-based schemes are only secure under the assumption of the security of the Rivest-Shamir-Adleman (RSA) public key algorithm [14].

In the work [15], the time-bound property based on the Lucas function is utilized to provide better performance and time efficiency and solve key update problems. The time-bound access control schemes are divided into two categories: the first one is based on tamper-proofed devices [16] and the second one [15], [17] is based on public values. Note that tamper-proof devices, which are collusion resistant but costly and unsuitable for the cloud, limit user convenience. Still, public values can be used efficiently in the cloud due to broad network access. Both public values and the user's own key are used to derive the secret key of lower classification levels. Thus, the numbers of the public values are critical to measuring the efficiency of the key assignment scheme.

In all other schemes proposed later, the main goal is to provide secure, dynamic, and efficient hierarchical key access control. The parameters that measure the efficiency and security of the hierarchical key assignment schemes in literature are as follows:

- 1) The amount of private and public information: The amount of information assigned to each class to derive the secret key. The data owner distributes private information to classes so that only users of desired classes have access. On the other hand, the data owner publishes all public information to all classes following the predetermined hierarchy.
- 2) Complexity of key derivation: The number of operations or computational requirements/cost needed for key derivation must be minimal and tolerable.

- 3) Complexity of key updates due to dynamic changes in the hierarchy structure: The scheme allows the deletion and insertion of classes in the hierarchy without the need for redistribution of any private information. In addition, the computational requirements needed for key updates have to be minimal and tolerable.
- 4) Resistance to collusion attacks known as collaborative attacks or key recovery (KR) attacks: The derivation of the secret key of each class has to be protected against any coalition of users belonging to the lower classification levels, and KR_s denotes such a secure scheme.
- 5) The state of key indistinguishability secure denoted by KI_s : The attacker should not distinguish between the secret key and a random string of the same length. If the scheme is KI_s , it means that it also provides key recovery security KR_s , but not vice versa [17].
- 6) Resistance to privilege creep problem: When the clearance 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, the scheme must guarantee both forward and backward secrecy.

In 1983, Akl and Taylor proposed a hierarchical access control scheme in a system to manage the key and to solve the hierarchical multi-group management and data sharing problem [2]. According to the proposed scheme, the computer (or communication) system users are divided into some disjoint sets $U_1, U_2, U_3, \dots, U_n$. Note that the relationship between classes is just a poset hierarchy. A security level is used to define each of the U_n and users in U_n can access data held by users in the same U_n or lower security level, while the vice versa is not allowed. The proposed scheme presents a solution to a hierarchical access control problem. The scheme can be useful in a secure distributed system. On the other hand, the approach does not completely solve more general multi-level security issues. This scheme can't be adapted flexibly and dynamically to the security policy determined by the data owner. The other problem with this method is that users can use the key permanently at a higher security level. A substantial amount of storage and communication is consumed due to the need to renew the keys periodically and redistribute them to the users. According to [17], the scheme expensively performs key derivation and only provides security regarding the KR . Due to the large number of keys held by each user, the scheme becomes inefficient as the number of users increases [6]. In addition to this, a large amount of storage for each security class to store public information is required [8], [13].

To control access to the data within a group of users ordered in a hierarchical structure, the improved scheme of [2], called a canonical assignment, is proposed to mitigate the amount of storage needed for public information, especially when the number of classes in the hierarchy is large [15]. However, the need for a large amount of storage is not eliminated [13]. Any member of a group can access the data of lower-class group users because they can generate lower-level group users' keys using their own key and the scheme

also provides security against collusion attacks.

In 1988, the work [7] proposed a tree hierarchical scheme based on symmetric-key cryptography in which security classes are organized as rooted-tree, which is a special instance of poset hierarchy. Using the iterative method of one-way function, which provides an efficient method to compute pre-images, the key belonging to the lower security class in the sub-tree can be generated. The most important innovation in this work is that new security classes can be inserted without changing keys for existing classes. 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 brings a substantial burden, especially for distributed and large infrastructures. In addition, there is also no need for extra public parameters for the key derivation. The main drawback of [7] is the computational overhead during deriving keys, especially when the key of the lowest class needs to be created by the root of the tree. The scheme is only implemented in a tree hierarchy, and it is not practical in trees greater than ten security levels.

In 1990, Harn and Lin proposed a similar approach to [2] but followed a bottom-top key derivation policy [8]. Unlike the works [2], [6], new security classes can be inserted without changing all keys. The storage need for public parameters for security classes is much less than [2], [6]. In addition, the scheme is more efficient in memory utilization in comparison with [2].

In 1993, Chang et al. [9] and Liaw et al. [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].

In 1993, Liaw and Lei [12] proposed an optimal heuristic algorithm for assigning cryptographic keys applying a top-down design approach in a tree structure for multilevel data security. 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], the algorithm can only be used in a tree structure.

In 1997, the work [13] modified the algorithm [12] to be used in poset hierarchy. Thus a user at a higher level can obtain the keys of other users at lower levels from his own cryptographic key. This is a one-way function, and the opposite direction is not allowed. The users collaborating at a lower level of the hierarchy would not obtain a higher level key for which they are not entitled. Thus the collusion, namely collaborated attacks are also prevented.

In 2002, Tzeng [15] proposed a time-bound cryptographic key assignment scheme inspired by [2] to prevent the key from being used continuously by members of higher level class C . According to the scheme, any user of a class C can only be a member of C for a certain period of time. A user in C_i can only 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 authorized users in a hierarchy with optimal bandwidth, and a user can only obtain data that he is granted access to. On the other hand, unauthorized users cannot obtain any data by listening to the broadcasting. In

addition, a user can hold encrypted data for only a period of time. A higher classified user can grant a privilege to another user to disclose the encrypted data, which ensures flexibility. The scheme is independent of the number of classes in the hierarchy, and this property did not exist in the previously proposed key assignment schemes based on poset hierarchy. However, the scheme is not efficient as expected since the users must always keep the keys in their hands to access the authorized data for a certain period of time. While the scheme 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 [16]. In addition, the scheme has been proven to be insecure against a feasible and efficient collusion attack if three users conspire to gain access to the keys [19].

In 2004, Chien [16] proposed an efficient time-bound hierarchical key assignment scheme inspired especially by [2] and [15]. The paper proposes to improve the time-slot-based key assignment scheme [15] by assigning distinct cryptographic keys to all to solve both implementation cost and performance issues in hierarchical key infrastructures. The scheme is based on a tamper-proof device that only performs simple arithmetic operations and is inaccessible even by its owner. There is a Trusted Agent (TA) and also a secure one-way hash function $h(\cdot)$. It is economically infeasible to derive the secret key from the public value. Users at lower security levels cannot obtain the key of higher security levels, so the scheme is resistant to collusion. No user can derive any key beyond the authorized time slots. In comparison with [15], the scheme appears much more efficient based on performance analysis. It needs a low-cost tamper-proof device that supports little storage space and simple operations without public-key cryptography and has little computational complexity. However, the scheme has been proven to be insecure against a feasible and efficient collusion attack if three users conspire to gain access to the keys [19], [18].

In 2006, the work [20] 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 requirements are minimal, and key updates are easier, but public data is quite high. In NBKAS, whose security relies 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 solve the problem of key updates/changes. 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 collusion-free which means that no combination of users can derive keys where they do not have authorization.

In 2009, Atallah et al. [21] proposed dynamic and efficient key management for hierarchical key access control. The scheme's main goal is to find a solution for the access control and key management problem in hierarchical infrastructures. The scheme works with random access graphs. Only hash functions are used for a node to derive the descendant's key from its own key. The number of linear bit processes limits key derivation by a node of the descendant's key, and the class consists of a single key associated with that class. Similar to [21], the scheme in the work [22] is also suitable to the dynamic changes of classes in the hierarchy, such as deletion and insertion of classes. The formal security analysis of the schemes so far has been made, and KI_s and KR_s notions first came up with [21]. The scheme is secure against chosen-plaintext attacks, and security is based on pseudo-random functions and an additional symmetric key encryption scheme. In the worst case, the key derivation step can require the implementation of $O(n)$ hash functions, where n is the number of nodes in the graph. It is more efficient than predecessors because of its dependency on interpolating polynomial, costly modular exponentiation operations, additional encryption. However, according to work [17], each user has to store a maximum of three private secrets, and the amount of public data inversely affects the complexity of key derivation. According to the study, [23], the number of public information increases with the number of edges on the graph and with the number of classes. In addition, as the number of levels between classes increases, the cost of key derivation increases linearly.

To solve the access control problem in the hierarchy, elliptic-curve cryptography-based hierarchic key assignment schemes were proposed [24] – [26]. In [24], the number of access control policies depends on the number of encryption keys, and tamper-proof devices play a key role in this scheme which is slower than [15], [16]. It has been proven that the scheme [24] is not secure against collusion attack [27]. In addition, Das et al. [28] have proven that the scheme described in the work [25] is vulnerable to exterior root-finding attack. Furthermore, the method presented in [26] is insecure against collusion attacks [29].

In 2011, De Santis et al. [30], and in 2012, Hassen et al. [31] proposed their schemes with better performance than the previous ones. The schemes in [30] (TBEBF:time-bound encryption-based family and TBEBF:time-bound broadcast encryption-based family) support dynamic updates in the hierarchy without the need of redistribution of private information but at the expense of an increase in the amount of public information. The schemes achieve KI_s and also improve the method of [21] computational requirements needed for key derivation and key updates remarkably. Key derivation (TBEBF) requires symmetric decryption approximately as many as the number of levels in the hierarchy, and TBEBF requires complex symmetric decryption. The scheme's private key storage requirement is small as it needs only one key per class, but the amount of

public storage need grows linearly with the number of classes and edges in the graph. On the other hand, the scheme in [31] has very competitive storage, bandwidth, and computation overheads in comparison with previous ones.

In 2012, Ateniese et al. [17] designed two different time-bound hierarchical key assignment 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. They consider the security of schemes regarding both KI and KR by attackers. These schemes are provable-secure and can compute the keys of all lower classes more efficiently in the hierarchy. In other words, the key derivation procedures are very efficient as only one decryption, or one pairing evaluation is sufficient regardless of the number of levels in the hierarchy. In addition, without any private information that needs to be changed, only local changes to the public information are enough to update the hierarchy. According to the work [23], private information can be as large as the number of periods (public information is already large) as the schemes are based not only on the number of classes but also on the number of time periods.

In 2013, Freire et al. [23] designed schemes based on pseudo-random functions PRF and forward secure pseudorandom generators $FSPRG$ for arbitrary posets. The schemes also show a trade-off between key derivation efficiency and storage requirements of private information. The ultimate efficiency of key derivation depends on the longest depth of poset. In contrast, the amount of private information depends on the width of a poset, but the key derivation efficiency is relatively better than others. The schemes do not need public storage. In addition, the updated KI_s notion of [21] provides stronger security SKI_s than all schemes so far.

In 2016 [32], hierarchical key access scheme based on linear-geometry organizes the sets of a user in a hierarchical order and divides the users into different disjoint groups called security classes to ensure different access privileges for each class. To derive the key at the descendant security level, the public vector of the user's level and the private vector at the ancestor security level can be used together. Without the need for iterative computation, the key of the descendant security level can be directly derived by the ancestor security level. The scheme only needs to compute the vector multiplication and values of the pseudorandom function, which causes very little computational overhead. The scheme also achieves SKI_s . Although the size of the public information in this scheme is slightly larger than the others, there is a balance between computation cost and storage space. The scheme provides an efficient key management solution that can serve as flexible and fine-grained hierarchical access control in cloud computing to address potential changes in the hierarchy with light computations in the finite field. However, the main drawback is the amount of public information compared to others, and there is a trade-off between computation cost and storage space. The ultimate overhead of every class might not be tolerable and efficient eventually. In addition, if there is a change in the hierarchy, the data owner must compute and publish a new public matrix. The other disadvantage is that the matrix should satisfy certain properties to establish the

relationship between the number of classes in the hierarchy and the public information, especially for rekeying.

There are two types of hierarchical key assignment schemes; indirect access schemes and direct access schemes. In indirect access schemes such as [23], [30], [31], the secret key of the child class can be derived from it by calculating all keys on the path to the child class. In direct access schemes such as [22], [25], [29], [32], it only requires one computational task to derive the secret key of the child class. However, their disadvantages are that they are not secure enough [28], [35], [36] and a high overhead task [29]. Therefore, there is still plenty of room 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.

3 PRELIMINARIES

3.1 Inner Product Space

Various applications of linear algebra have been presented for the last century [33]. Matrices, eigenvalues, linear systems are indispensable tools in computational science, especially in artificial intelligence and machine learning-related research [34]. 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. 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 Gram-Schmidt orthogonalization process is briefly demonstrated. Finally, we explain the last step to find orthogonal projection (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 Gram-Schmidt (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 orthogonal projection for a vector subspace. An orthogonal projection 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 orthogonal projection 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 access scheme.

3.2 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 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 . 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 orthogonal projection of f on W will be the same. As long as a basis of W is known, the unique orthogonal projection can easily be computed via the Gram-Schmidt orthogonalization process. Figure 3 depicts the idea of OP on an inner product space.

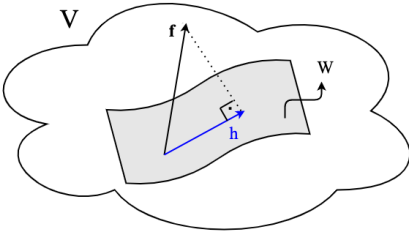


Fig. 3. **Orthogonal Projection:** The closest vector h to f is the orthogonal projection of f onto W . In other words, h is the nearest vector to f in W if and only if $(f - h)$ is perpendicular to all vectors in W .

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 orthogonal projection of a vector f in V .

4 THE PROPOSED SCHEME

Our proposed scheme is based on the following rules.

Rule 1. G_1 is the root class and the most privileged group, and $c \geq 0$ is the number of classes in the hierarchy.

Rule 2. For any two classes $G_i, G_j \in V$, $G_i \leq G_j$ means that any user in G_i can access its own secret key K_i and also K_j belonging to G_j whose security clearance 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 G_i has to be one of the parent classes of G_j and if they are not in the parent-child relationship like in Figure 1, then K_j should not 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 change in the hierarchy. For example, in this respect 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. All vectors v_i that make up the elements of each S_i must 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 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 come together and combine the information they have, they can not generate the subspace W_2 and 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 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 clearance level group G_3 (the parent of these three class) has the set S_3 containing $v_{10}, v_{11}, \dots, v_{21}$ and other 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 of 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 clearance level than G_3 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 . The above illustration can also be adapted efficiently to an N-class hierarchy by applying an LRBU policy.

4.1 Preparation phase

The data owner:

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

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

Step 3. Identifies subspaces W_1, W_2, \dots, W_c . Notice that the selected V structure allows the data owner 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 clearance level) for predetermined subspaces W_i to distribute each class/group G_i for $i = 1, 2, \dots, c$ in 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}_q to distribute to users U_i in the group G_i .

4.2 Key distribution phase

At this stage, the data owner shares the public and private information with the relevant groups G_i that enable each member U_i to derive the corresponding key K_{G_i} .

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 all the members.

Step 2. Determines all corresponding basis sets S_i for each W_i assigned to the class G_i in the hierarchy.

Step 3. Distributes basis sets S'_i derived from the base set S_i to each member of the group G_i .

4.3 Key derivation phase

Note that once the users receive their own basis (public and private information), they can form an orthonormal basis for W_i by applying the Gram-Schmidt method to their basis and then apply orthogonal projection (OP) of f onto W_i to extract the secret K_{G_i} . (Figure 4).

To derive the corresponding secret key K_{G_i} , any user in the hierarchical structure:

Step 1. Applies Gram-Schmidt orthogonalization operation to the given basis, namely computes corresponding orthonormal basis. (See 3.1)

Step 2. Runs OP in order to obtain the corresponding key K_{G_i} . (See Figure 3)

To derive the key for the class G_i , each user computes:

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

Notice that the K_{G_i} for each group G_i requires having a basis for the corresponding subspace W_i . On the other hand, for a child to have its parent's key, it also needs the parent's private information, which will be a vector and is not feasible for the children 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 2, it is not feasible to obtain W_1 from W_2 . In fact, there are many subspaces of \mathbb{R}^n of three-dimension, and if one knows only two basis elements of it, 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 and performance of our scheme are based.

4.4 Dynamic update phase

The scheme allows the insertion and deletion of classes in the hierarchy without the need to redistribute and change any public information f used by any G_j for both cases below. The insertion and deletion of any class G_j have to be following security and LRB policy.

Insertion. If a new group G_j is inserted into the hierarchy, a new basis set S_j is given to this new group as private information concerning its clearance level by the data owner. Then, all related parents with G_j will have the updated basis set, including the basis elements of the S_j . Note that there is no need to update any child's basis set.

Deletion. If any G_j which is a parent of some children, is removed from the hierarchy, all children of the removed parent will be linked to one higher parent class and the new related parents' basis sets. However, if the deleted group is at the bottom of the hierarchy, there will be no change in linkage. In addition, if the data owner removes or deletes any G_j in the hierarchy, the basis set of G_j will also be removed from the higher parent classes to ensure efficiency and reduce disk space requirement.

5 IMPLEMENTATION RESULTS, PERFORMANCE AND SECURITY ANALYSIS, AND COMPARISON

We implemented the proposed hierarchical key access mechanism on a computer with Windows 10 operating system running on Intel Core i5-6200 CPU 2.30 GHZ X-64 bit processor. It has 16 GB of memory, and we use JAVA programming language, and Eclipse integrated development environment (IDE).

The extracting process of a secret key K_i requires first finding an orthonormal basis described in Algorithm 2 (Gram-Schmidt orthogonalization process). Upon constructing an orthonormal basis, the method employs orthogonal projection, which outputs the K_i where Algorithm 1 stands for the key extracting process. 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.

5.1 Performance Analysis

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

Public and private storage needs, key derivation, and key update overhead are critical metrics to gauge the efficiency of our scheme. Note that the number of classes G in the hierarchy is c . The maximum dimensions of a subspace associated with any class at the bottom in the hierarchy are denoted by b .

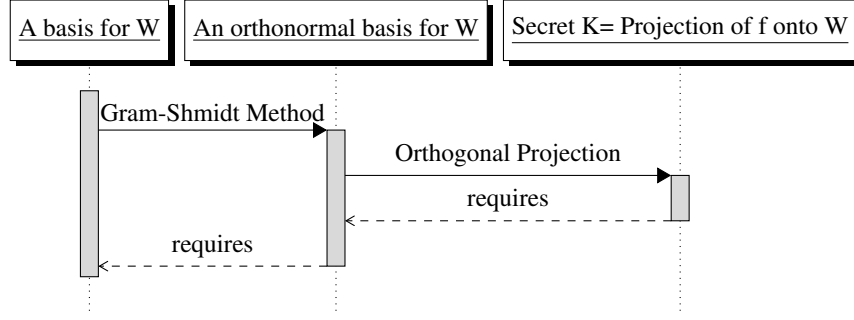


Fig. 4. The Needed Steps of the Secret Key Derivation

Algorithm 1:

Input:

- 1 A subspace W and its basis $\{b_1, b_2, \dots, b_n\}$ where $W \subset V$ infinite dimension
- 2 $U_{ij}, i = 1, 2, \dots, h, j = 1, 2, \dots, t$ where h is the number of groups and t is the number of users in h^{th} group
- 3 $k_i, i = 1, 2, \dots, s$ where s is the dimension, $k_i < m$ and $k_i < n$

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** h **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** t **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 $K_{ij} \leftarrow \sum_{i=1}^n \langle f, w_i \rangle \cdot w_i$
-

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 key K_{G_i} .

Private information storage need. Each member of the class needs its own basis S_{U_i} to derive the K_{G_i} . 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 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

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

child classes at the bottom have only one private information, whereas the classes at the bottom have a maximum private information need of b .

Key derivation cost. Let K be the secret key of a class. The K is extracted from users' private information and requires access to data. The extraction process involves Gram-Schmidt orthogonalization operation and projection of the public vector f on the subspace generated by the users' private basis elements. The actual cost depends on the size of the basis and the defined inner product on the universal set V . To compute the cost of these steps, we fixed the group G and the corresponding subspace, which W denotes. Let n be the dimension of the subspace W , which is assigned to the group G . We will address the cost of extracting the key for a user in three categories: the required number of inner product I , multiplication M , and division D operations.

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

- 1) Gram-Schmidt 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) Gram-Schmidt 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) Gram-Schmidt 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 cost of deriving the secret key K for a user is $O(n^2)$ where the ultimate cost is

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

Key update. The scheme allows the insertion and deletion of classes in the hierarchy without the need to redistribution any public information. In addition, computational requirements redistribution of any private information needed for key updates (if there is an insertion) is minimal.

Change in the hierarchy. In case of a change (insertion or deletion) in the number of users and classes, there is no need for extra private or public information, so the key derivation cost will be constant when any class is deleted. As seen in Figures (5-7), key derivation for relevant parent classes will increase linearly only with the insertion of a new child class as the dimension of the subspaces increases. Note that the change in the hierarchy might also cause to privilege creep problem, and our scheme brings a solution for this with a little extra storage need and key derivation cost. (see 5.2).

We implement Algorithm 1 and Algorithm 2 while selecting real vector space \mathbb{R}^n over real numbers to observe the real-time cost of the key derivation process. The inner product then is chosen to be the dot product. The dimension of vector subspaces vs. maximum total key derivation cost (milliseconds) is depicted in Figures (5-7) below. The key derivation cost increases linearly as the dimension of vector subspaces increases. When calculating the key derivation cost for any user U in the group G , only for the 11th and 12th steps of Algorithm 1 have been considered. Namely, these steps are directly related to the actual key derivation cost after the startup phase (up to 10th step) of the algorithm is completed.

5.2 Security Analysis

KR_s . Because each class in the hierarchy has a distinct basis S , any child class, and any same level class cannot derive any private 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 provides at least KR_s .

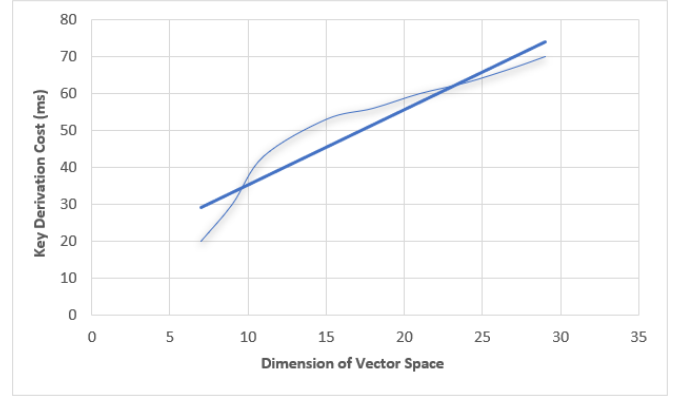


Fig. 5. Key derivation cost (milliseconds) up to dimension 30

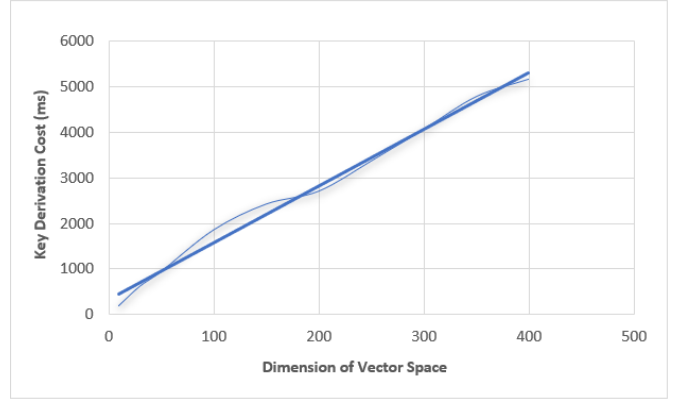


Fig. 6. Key derivation cost (milliseconds) up to dimension 400

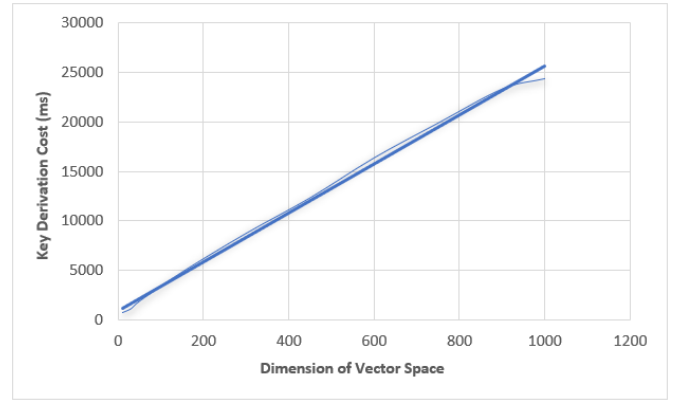


Fig. 7. Key derivation cost (milliseconds) up to dimension 1000

KI_s . Our scheme's security is based on 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 the security of our scheme has provided KI_s .

Resistance to privilege creep problem. If there is a change in the hierarchy, which means that the clearance 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 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 group gets extra basis element v_i distributed to all users in this group and relevant parents (inserting only one element into the basis of the relevant classes). In this way, the maximum private storage will increase by one as relevant space's dimension increases by one, and the key derivation 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 storage cost.

5.3 Comparison with Other Schemes

Table 1 reveals a comparison between our scheme 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 1.

In Akl and Taylor based schemes [2], [6] – [13], [15], [16], 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 satisfy KR_s and KI_s under the RSA assumption or random exponent. KR_s and KI_s notions, in other words, a provably secure scheme first came up with [21].

In Atallah et al. [21] and De Santis et al. [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 cost is directly related to the path length between classes. Namely, the cost grows if the path length between classes increases. From the security perspective, these schemes are KI_s and CPA_s .

In D'Arco et al. [14], it has proven that Akl and Taylor-based schemes [6] – [13], [15], [16] (KR_s by default) can only be KI_s under RSA assumptions RSA_s . But, to achieve KI_s property, the additional public storage need and key derivation cost must be met, which will be a quite multiple of the bit length l of the key.

In Ateniese et al. [17], the key derivation is very efficient regardless of path length between classes. The main drawbacks are related to storage needs. For the first proposed scheme, both the number of classes and the number of time periods are parameters to define the need for public information. For the second proposed scheme, the number of time periods is the critical parameter to define the need for private information.

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

In Tang et al. [32], the direct access scheme without the need for iterative computation addresses the potential changes in the hierarchy with light computations as well as achieves SKI_s . But, the main drawback is the amount of public information compared to others, and there is a

trade-off between computation cost and storage space. Each class needs to compute two times M (computational time of modular multiplication), and one time A (computational time of modular addition) over F_q to derive its K . On the other hand, each class needs to compute twice the value of F , four times M , and two times A over F_q . Thus, the ultimate overhead of each class might not be tolerable, which makes the scheme inefficient. In addition, if there is a change in the hierarchy, the data owner needs to compute and publish a new public matrix.

On the other hand, our scheme is a flexible and fine-grained hierarchical direct access control scheme and does not need other classes to derive the secret key. Key derivation cost n^2 . Our scheme's public and private storage needs are tolerable and guarantee 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.

6 CONCLUSION

In this work, the inner product spaces have been applied to construct a hierarchical key assignment scheme employed in various situations, especially for any cloud infrastructures.

Public and private storage requirement, key derivation, and key update overhead, 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 schemes in the literature. For instance, public and private storage needs are among the main burdens for the data owner 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 derivation and updating keys is also provided in our scheme.

From 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. This proposal also provides 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. Varadharajan, 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.

TABLE 1
Comparison with Previous Schemes

Schemes	Dynamic	Private Storage	Public Storage	Key Derivation	Security Type	Security Assertions
Akl and Taylor based [2] [6] – [13], [15], [16]	No	One	$c.t$	t_{exp}	KR_s	N/A
Atallah et al. [21]	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 [17]	Yes	One	$c^2.t^3$ (max)	DT_{se}	KI_s	CPA_s
Ateniese et al. TLPBC [17]	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
The proposed scheme	Yes	One (min) b (max)	One	n^2	SKI_s	$LICV$

Notation:

c : number of classes G in the hierarchy
 t : total number of time period
 w : partially ordered set (poset) width
 e : number of edges in the hierarchy
 l : bit length of the key
 t_{exp} : exponential computational time over a large group ($D+E$).
 DT_{se} : decryption time using a symmetric key encryption scheme
 DT_p : decryption time using pairing evaluation
 T_{prf} : time of pseudo-random functions (PRF) evaluation
 T_{prg} : time of pseudo-random generator (PRG) evaluation
 h : path length between the classes in hierarchy
 n : dimension of the subspace W which is assigned to a group G
 CPA_s : secure against chosen plaintext attack
 RSA_s : secure under the RSA assumption
 $FSPRG$: forward-secure pseudorandom generator
 b : maximum dimension of a subspace associated with any class at the bottom in the hierarchy
 $LICV$: linearly independent chosen vectors

- [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. Harn, 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.
- [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] 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.
- [16] 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
- [17] 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.
- [18] 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
- [19] 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.
- [20] 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.
- [21] 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.
- [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. 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, 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] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web", Technical Report, Stanford InfoLab, 1999.
 - [34] 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.
 - [35] 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.
 - [36] 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.



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.



Enver Ozdemir (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.



Cyber Security, Network Security, Cloud Computing, and Cryptography.

Baris Celiktaş 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