IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 26 February 2024, accepted 13 May 2024, date of publication 16 May 2024, date of current version 24 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3401844

RESEARCH ARTICLE

A Hierarchical Key Assignment Scheme: A Unified Approach for Scalability and Efficiency

IBRAHIM CELIKBILEK[1], BARIS CELIKTAS[2], (Member, IEEE), AND ENVER OZDEMIR[1], (Senior Member, IEEE)

[1]Cybersecurity Engineering and Cryptography Department, Informatics Institute, Istanbul Technical University, 34469 Istanbul, Turkey
[2]Computer Engineering Department, Işık University, 34398 Istanbul, Turkey

Corresponding author: Ibrahim Celikbilek (celikbilek20@itu.edu.tr)

ABSTRACT This study introduces a hierarchical key assignment scheme (HKAS) based on the closest vector problem in an inner product space (CVP-IPS). The proposed scheme offers a comprehensive solution with scalability, flexibility, cost-effectiveness, and high performance. The key features of the scheme include CVP-IPS based construction, the utilization of two public keys by the scheme, a distinct basis set designated for each class, a direct access scheme to enhance user convenience, and a rigorous mathematical and algorithmic presentation of all processes. This scheme eliminates the need for top-down structures and offers a significant benefit in that the lengths of the basis sets defined for classes are the same and the costs associated with key derivation are the same for all classes, unlike top-down approaches, where the higher class in the hierarchy generally incurs much higher costs. The scheme excels in both vertical and horizontal scalability due to its utilization of the access graph and is formally proven to achieve strong key indistinguishability security (S-KI-security). This research represents a significant advancement in HKAS systems, providing tangible benefits and improved security for a wide range of use cases.

INDEX TERMS Hierarchical key assignment, access control, closest vector problem, inner product space, access graph, strong key indistinguishability.

I. INTRODUCTION

Access control involves the authorization and restriction of assets based on business and security objectives. Access control processes refer to the need to control subject-to-object access. The access control problem arises from problems with authorized users accessing critical data. Hierarchically, users are organized into different groups (classes) to mediate access between subjects and objects. Some subjects have more privileges than others, and some objects contain more critical data, thus requiring increased security measures. For example, in the context of database access, consider a scenario involving two distinct roles: Database Administrator and Marketing Analyst. The Database Administrator, represented by X, has extensive permissions for the financial database. X possesses both read and write access to all tables, allowing him to retrieve data and change them as needed when generating financial reports. However, the situation differs for Y, a Marketing Analyst. Y's role is limited to marketing-related tasks and responsibilities. Consequently, she has been assigned read-only access specifically to marketing-related data in the database. This access restriction prevents her from accessing any financial data or generating financial reports.

Implementing and managing access control mechanisms involves establishing and overseeing systems that regulate user access to objects based on defined policies. This process ensures that only authorized entities are granted appropriate permissions to access specific objects or perform certain actions. Effective implementation and management of authorization mechanisms are crucial for maintaining security and controlling data integrity within an organization's information systems.

The main objective of an access control system is to protect system resources from access by unauthorized users. This is achieved by implementing policies and technologies that

The associate editor coordinating the review of this manuscript and approving it for publication was P. Venkata Krishna.

restrict access to systems and resources to those authorized to do so. This is critical to maintaining the confidentiality, integrity, and availability of sensitive information. Access control systems can range from simple password-based mechanisms to more complex solutions with biometric authentication and cryptographic key management. They are an essential part of an organization's security framework, helping to prevent data breaches, protect against insider threats, and ensure compliance with various regulatory requirements. By effectively managing who can access which resources and under what conditions, access control systems play an important role in securing an organization's digital and physical assets.

Hierarchical key assignment schemes are a crucial component of access control, aiming to offer a secure and efficient method of managing keys in a structured environment. By implementing hierarchical key management, we ensure the confidentiality, integrity, and availability of data in a manner that is both stringent and adaptable. This approach allows only authorized individuals to access the data, effectively safeguarding it.

HKASs involve defining public information for the entire scheme, along with private information specific to each class within the hierarchy. Utilizing this blend of private and public information, a user in a given class can effectively compute their own unique secret key and the keys for classes at lower levels in the hierarchy. This structure ensures controlled access while maintaining the integrity of the hierarchical access model.

In particular, hierarchical key assignment schemes play a vital role in various fields where access control is of utmost importance, and hierarchical structures naturally exist. This includes domains like cloud computing, organizational data access, healthcare systems, multilevel databases, the Internet of Things (IoT), and the coordination of drone swarms. For instance, in the context of cloud computing, due to the inherently multi-tiered structure of cloud access and the diverse responsibilities of cloud users and administrators, employing a hierarchical key assignment method is well suited. Hierarchical key assignment schemes offer specific advantages tailored to cloud computing, such as scalability, enhanced security, efficient key management, and adaptable access delegation. This comprehensive approach ensures that the data is protected effectively and efficiently.

This work introduces a comprehensive generic hierarchical key assignment scheme that is applicable to cloud computing environments and emphasizes scalability, flexibility, cost-effectiveness, and high performance. If implemented for the cloud, it can address critical concerns related to data access policy integration, potentially offering a valuable solution for organizations considering the transition to cloud-based storage services.

II. RELATED WORKS

The study by Akl and Taylor [2] proposed a cryptographic method to manage access control within hierarchical structures. This approach organizes users into distinct, non-overlapping classes, denoted as $C_1, C_2, \cdots, C_n$, and arranges these classes in a partially ordered set (poset), assigning specific security levels to each class. Within this poset, users belonging to a particular class are granted access to data controlled by users in the same or lower-level classes. Importantly, the scheme's design effectively prevents users at lower security levels from accessing data in classes of higher security levels, thereby ensuring a secure access control mechanism. Despite its advantages, the scheme presents certain challenges. Users at higher levels must manage numerous cryptographic keys, leading to complicated key management problems. Furthermore, the model cannot solve broader security problems at multiple levels, such as the protection and management of classified information.

Following the seminal work of Akl and Taylor, numerous hierarchical key assignment schemes have been proposed. These include [3], [4], [6], [7], [8], [9], [10], [11], [12], [13], [14], [17], [18], [19], [21], [22], [24], [25], [26], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [41]. Our analysis places particular emphasis on a select group of schemes deemed especially significant, which are [22], [26], [29], [30], [31], [32], [33], [34], [35], [36], [37], [39], and [41]. These schemes are integral to our comparison table and are thoroughly and comparatively analyzed in subsection V-C.

In addition, in the following, brief summarizing information will be provided about some other schemes selected from those referenced above but not included in the comparison table. These additional details will contribute significantly to understanding the wide diversity in the literature and assessing the originality of our work.

In [10], a novel heuristic algorithm for cryptographic key allocation in tree structures is introduced, improving multilevel data security and minimizing key sizes. In this scheme, which employs a top-down approach, user groups are represented through a tree structure [11]. The scheme requires further development to enable the addition of new users with minimal key changes. The study in [11] allows the algorithm introduced in [10] to be used alongside a poset structure, enhancing its application beyond the original tree-based framework. In this scheme, higher-level users can derive keys of lower-level users from their own cryptographic keys; however, the reverse is not allowed. Both approaches effectively address the collusion (collaborative/key recovery) attacks, a prevalent issue in many past HKASs. According to [28], both schemes have trade-offs between the amount of public and private information and the complexity of the key derivations.

The work by [13] introduced a time-bound HKAS based on a poset for hierarchical structures, and also presented two practical applications for this scheme. The first application, secure broadcasting, entails broadcasting data exclusively to authorized users, ensuring that each recipient can access only the information designated for them. The second application, cryptographic key backup, involves creating a cryptographic

key backup system, which refers to the secure method of keeping cryptographic keys so that they can be recovered in case the original keys are lost, damaged, or otherwise compromised.

In this study, each class C_i is assigned a series of distinct keys for different time periods, denoted as $K_{i,t}$. This implies that a user will have access to the keys of their class or its subclasses during a specific period of time. Furthermore, it allows users to join a class for a designated duration. A user belonging to class C_i can compute $K_{i,t}$ in the time period specified by the start (t_1) and the end (t_2). Similarly, a user can also compute $K_{j,t}$ if $C_j \leq C_i$ and $t_1 \leq t \leq t_2$. The size of the private information that needs to be stored to derive class keys is independent of both the total number of classes and the size of the time period. However, the scheme is computationally inefficient because it involves expensive public key computation (operations) to derive class keys [18].

The work by [18] is based heavily on [13] and utilizes a tamper-resistant device for its construction. The number of public parameters, the computational complexity required for key derivation, and the implementation cost used by [18] have been significantly improved compared to [13]. Within a specified time period, a user from a higher class can derive the key of the lower classes from their own key. In [18], the cost of the key derivation is highly dependent on the complexity of the hash operations, while in [13], the cost of the key derivation increases directly proportional to the complexity of modular exponentiation and the calculations of the Lucas function. The studies [13] and [18] are considered insecure against collusion attacks according to [16] and [20] respectively.

The schemes designed in studies [21], [24], and [25] are based on elliptic curve cryptography (ECC), which generally offers similar security levels with smaller key sizes and lower computational cost compared to classic public-key cryptography such as RSA and others [21]. According to [30], the key derivation cost of the scheme [21] is lower than [25], and the scheme [21] is not secure against a compromising attack. The scheme in [24] is time-bound and the number of encryption keys is determined by the number of access control policies. Similar to [18], scheme incorporates a tamper-resistant device. In addition, according to [27], scheme is insecure against collusion attacks. As the number of classes in the hierarchy increases, the number of public keys used by [24] will be greater than those used by [13] and [18], and the scheme of [24] is slower than [18].

Our proposed scheme makes significant contributions to the literature in the following ways.

- We introduce and utilize the CVP-IPS as an extension of the standard CVP (CVP-Lattice). One of the most important advantages of the presented scheme is its effectiveness and the ability to be implemented on small devices such as mobile phones, smart cards, etc.
- There are two public keys pub=(f_1, f_2) for the entire scheme, and the dynamic update procedures KeyRevocate, KeyRollover, ClassReInitialize, ClassInsertion, ClassDeletion, UserInsertion, UserDeletion (see Algorithm 4) do not require changing the public keys.
- In our scheme, we define different basis sets of the same length for each class. Additionally, no public or private information is defined for the edges that constitute the access graph.
- A user who is a member of a class within the hierarchy can access both the secret keys of their own class and those of other classes below it in the hierarchy with a single lightweight computation (direct access scheme).
- All processes included in the scheme are fully presented mathematically and algorithmically.

A crucial aspect of our proposed scheme is its robust vertical and horizontal scalability, due to the access graph utilized. With a formal security proof, the suggested technique also achieves strong key indistinguishability security (S-KI-security).

III. PRELIMINARIES

A. INNER PRODUCT SPACE

An inner product space $\mathcal{V}$ is a vector space over a field $\mathcal{F}$, where $\mathcal{F}$ can be real numbers $\mathbb{R}$ or complex numbers $\mathbb{C}$. The space is equipped with an inner product, which is a function $< u, v >: \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{F}$ that maps each pair of vectors $u, v \in \mathcal{V}$ to a scalar in $\mathcal{F}$. This function must satisfy certain properties, including linearity in the first and the second arguments, conjugate symmetry, and positive definiteness. An inner product allows discussion of the orthogonality of vectors, projections, and other geometric concepts within a space [41].

The inner product space currently serves as the foundation for various cryptographic algorithms and protocols. It provides a useful mathematical framework for the development of cryptographic protocols and is a crucial instrument to ensure the security of contemporary communication systems. Inner product space-based cryptographic schemes that are widely researched include inner product encryption (IPE), inner product functional encryption (IPFE), inner product signature (IPS), linearly homomorphic encryption, cryptographic hash functions, bilinear pairings, public key encryption, attribute-based encryption, and so on.

B. CLOSEST VECTOR PROBLEM (CVP)

The Closest Vector Problem (CVP) is a computational problem in lattice-based cryptography. A lattice is a geometric structure consisting of points in an n-dimensional Euclidean space that are organized in a periodic and repeating pattern [15]. The CVP has several versions (variants) that arise depending on the constraints introduced, including search, optimization (or approximative), and decision CVP. Within any constant factor or even some slowly increasing subpolynomial function of dimension n, CVP is known to be approximately NP-hard to solve [1], [15]. Many cryptographic systems rely on CVP and its variants because of its hardness, including lattice-based public-key encryption, digital signatures, fully homomorphic encryption (FHE),

identity-based encryption (IBE), attribute-based encryption (ABE), post-quantum key exchange, etc.

CVP-IPS can be viewed as an extension of a specific instance of CVP-Lattice. It is important to note that with CVP-IPS, unlike the standard form of CVP-Lattice, the basis set is not open to the public.

Definition 1 (CVP-IPS): Let $\mathcal{V}$ be an inner product space, preferably of infinite dimension, and let W be a subspace of $\mathcal{V}$, without any specified basis. Given a vector $f \in \mathcal{V}$ such that $f \notin W$, finding a unique vector $f^* \in W$ that is closest to $f \in \mathcal{V}$ is called CVP-IPS, which provides information-theoretic security.

Given a basis $B = \{w_1, w_2, \cdots, w_n\}$ for the subspace W, the Gram-Schmidt algorithm can be employed to convert B into an orthogonal basis B' (also an orthonormal basis B''). For given f and B'', we can always find a unique $f^* \in W$ closest to $f \in \mathcal{V}$. The unique vector is calculated by the projection of f onto the subspace W;

$$f^* = proj_W(f) = \sum_{i=1}^{n} \langle f, w_i'' \rangle w_i''$$

where $\langle , \rangle$ inner product. On the other hand, if no basis set of the subspace W is known, we can not find f^*. It can be shown that CVP-IPS provides information-theoretic security since obtaining the subspace W with only partial knowledge (without all members of the basis set) is computationally infeasible.

IV. THE PROPOSED SCHEME

In this section, the general structure, mathematical basis, symbols, syntax, and phases of the hierarchical key assignment scheme are explained in detail.

A. GENERAL STRUCTURE AND DEFINITIONS

Assume that a system's users fall into a number of distinct sets (groups), and each group will be referred to as a class. Let the set of classes be $\mathcal{C} = \bigcup_{i=1}^{c} C_i$ with $c = |\mathcal{C}| \geq 1$ and C_1 be the most privileged (root) class in the hierarchical access structure.

In our scheme, a partially ordered set (poset) is used as the representation of the hierarchy. Assume that $(\mathcal{C}, \geq)$ is a poset and the meaning of $C_i \geq C_j$ in $(\mathcal{C}, \geq)$ is that C_i one of the ancestor classes of C_j, the users in C_i have a security clearance (classification) level higher than or equal to those in C_j. Any poset $(\mathcal{C}, \geq)$ can be represented by an access graph $G = (\mathcal{C}, E)$, where each class in $\mathcal{C}$ corresponds to a vertex in G. From the point of view of the access graph, there can be one or more directed paths from C_i to C_j, but the length of each directed path must be equal. For example (see Figure 1), there are two directed paths (C_1, C_2, C_5),(C_1, C_3, C_5) from C_1 to C_5 with the same length. In terms of our access graph, the notation $C_i \geq C_j$ means that there is at least one directed path from C_i to C_j.

Let us assume that an access graph is demonstrated in Figure 1. Each vertex (node) represents a distinct class C_i,

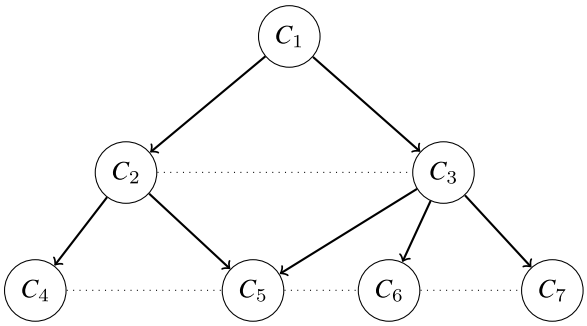

FIGURE 1. Access graph based on partially ordered set (poset).

which has a unique secret key K_i. In the access graph, the depth of a class C_i is its distance from C_1, and classes with the same depth have the same security level. We define two functions [26] $Anc(C_i)$, $Desc(C_i)$, the set of ancestors and descendants classes of C_i, as follows.

$$Anc(C_i) = \{C_j \in \mathcal{C} | C_j \geq C_i\}$$
$$Desc(C_i) = \{C_j \in \mathcal{C} | C_i \geq C_j\}$$

Let $\mathcal{V}$ be an inner product space, preferably of infinite dimension, over $\mathbb{R}$. Given that there are uncountably many n-dimensional subspaces in $\mathcal{V}$, we can generate a unique subspace $W_i = \text{span}(B_i)$ with $\dim(W_i) = |B_i| = n$ for each class C_i. Let us define $\mathcal{B} = \bigcup_{i=1}^{c} B_i$, $\mathcal{W} = \bigcup_{i=1}^{c} W_i$ such that $B_i = \mathcal{P} \cup S_i$, $W_i = \text{span}(B_i)$. We note that $B_i \neq B_j$, $W_i \neq W_j$ for all $i \neq j$ and $W_i \subset \mathcal{V}$ for $i : 1, \cdots, c$. The length of each basis set $B_i \in \mathcal{B}$ is the same, that is, $|B_i| = n$ for $i : 1, \cdots, c$. This is due to the fact that it guarantees that the method we provide is scalable and efficient, unlike top-down key assignment schemes. Additionally, the computational cost of key derivation remains constant regardless of how many classes there are in the hierarchy.

Each basis set $B_i \in \mathcal{B}$ is a union of two subsets $B_i = \mathcal{P} \cup S_i$ where $|S_i| = s$, $|\mathcal{P}| = n - s$, $S_i \neq S_j$, $S_i \cap S_j = \{\}$, for all $i \neq j$. It can be stated that the linear independence of all vectors in the set B_i and $\mathcal{S} = \bigcup_{i=1}^{c} S_i$ holds true. If descendants classes of C_i or other classes that are at the same security level as C_i collaborate to gain access K_i, they would have to learn a unique S_i that belongs to C_i which is a computationally infeasible task.

Let Γ be a set of access graphs corresponding to partially ordered sets (hierarchies); a hierarchical key assignment scheme can be defined as follows.

Definition 2: A hierarchical key assignment scheme for Γ consists of two polynomial-time algorithms **(Gen, Der)** such that:

1) **Gen**$(\mathbf{1}^\rho, \mathbf{G})$ is a probabilistic information generation algorithm, executed by the data owner. It takes as input a security parameter 1^ρ(i.e., written in unary) and a graph $G = (\mathcal{C}, E) \in \Gamma$ and produces:
   - For each class $C_i \in \mathcal{C}$
     - a private information $B_i = \mathcal{P} \cup S_i$
     - a key $K_i \in \{0, 1\}^\rho$

- A public information $pub = (f_1, f_2)$

The output of $Gen(1^\rho, G)$ is denoted as (B_*, K_*, pub), where B_* represents private information, and K_* is a secret key.

2) **Der**$(\mathbf{1^\rho}, \mathbf{G}, \mathbf{pub}, \overbrace{B_i, \cup_{C_j \in Desc(C_i)} S_j}^{\mathbf{u_{ix}-secrets}})$ is deterministic key derivation algorithm, executed by u_{ix} (x^{th} user of C_i). It takes as input 1^ρ(the security parameter), a graph G, public information pub, and a user $u_{ix} \in C_i$. The last parameter of the algorithm is optional. The return values of the algorithm are as follows.

$$Der(1^\rho, G, pub, B_i) = K_i$$
$$Der(1^\rho, G, pub, B_i, S_j) = K_i, K_j \text{ iff } C_i \geq C_j$$
$$Der(1^\rho, G, pub, B_i, S_j) = \perp \text{ (rejection), others.}$$

The scheme we suggest assumes that group users will receive private key components in a secure manner. In this study, we focus on how $u_{ix} \in C_i$ in a hierarchical structure can securely and effectively calculate both the K_i and the K_j (iff $C_i \geq C_j$). In this respect, the algorithm for the distribution of key components to relevant users is not included in the formal definition of our HKAS.

B. SYSTEM-PREPARATION AND DISTRIBUTION PHASE

The preparation and distribution processes for the system are thoroughly described in Algorithm 1. In addition, Figure 2 offers a concise overview of the preparation and distribution phase. The data owner (u_{11}) is a member of class C_1 (root, most privileged) and all operations in Algorithm 1 can only be executed by u_{11}. Public information, consisting of two vectors, pub=(f_1, f_2) produced by Algorithm 1 (stat. 10, procedure fVector) is set to public for all users in the access hierarchy (u_{**}). The scheme we propose does not include any key distribution center (KDC) or trusted third party (TTP).

To create the set $\mathcal{P}$, Algorithm 1 randomly picks $(n - s)$ linearly independent vectors from $\mathcal{V}$ (syntax: $\xleftarrow[(n-s)]{\$}$), see Algorithm 1 (stat. 3).

For each class $C_i \in \mathcal{C}$, see Algorithm 1 (stat. 5-9):

- A unique basis set B_i is constructed (stat. 5,6) and securely distributed to all users of the C_i (u_{i*})(stat. 8).

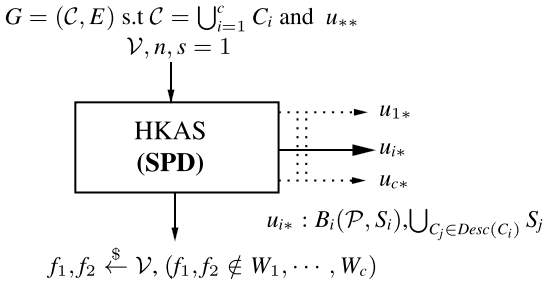

FIGURE 2. Illustration of algorithm 1, SPD: system preparation and distribution.

- A unique S_i is securely distributed to users of the classes in the $Anc(C_i)$, see Algorithm 1 (stat. 9, procedure DistributeSecret).

At the end of the distribution process, each user of class C_i has the sets $B_i(\mathcal{P}, S_i)$ and $\bigcup_{C_j \in Desc(C_i)} S_j$. Furthermore, since $u_{11} \in C_1$ and $C_1 \geq C_i$ for all $i : 2, \cdots, c$, u_{11} will have $\mathcal{P}$, $\mathcal{S} = \bigcup_{i=1}^{c} S_i$. Let us symbolize the set $\mathcal{S}$ owned by u_{11} as $\mathcal{S}^{u_{11}}$ to use in Algorithm 4 (dynamic update process).

$$\mathcal{S}^{u_{11}} = \{C_1 : S_1, C_2 : S_2, \ldots\}, \mathcal{S}^{u_{11}}[C_i] = S_i$$

where $\mathcal{S}^{u_{11}}$ is a dictionary (or mapping), that is, $\mathcal{S}^{u_{11}}$ is an unordered set that maps C_i(key) to S_i(value).

Algorithm 1 System-Preparation and Distribution Processes

Input:
1 $G = (\mathcal{C}, E)$ s.t $\mathcal{C} = \bigcup_{i=1}^{c} C_i$ and u_{**}
2 $\mathcal{V}, n, s = 1$(default)
Proc DistributeSecret(C_i)**:**
  for C_j *in* $Anc(C_i)$ **do**
    distribute securely $S_i \rightarrow u_{j*}$
Proc fVector()**:**
  $(f_1, f_2) \xleftarrow{\$} \mathcal{V}, (f_1, f_2 \notin W_1, \cdots, W_c)$
  publish (f_1, f_2)
Initialization:
3 $\mathcal{P} \xleftarrow[(n-s)]{\$} \mathcal{V}$
4 **for** i = *1 to* c **do**
5 $S_i \xleftarrow[s]{\$} \mathcal{V}$
6 $B_i = \mathcal{P} \cup S_i$
7 $W_i \leftarrow \text{span}\{B_i\}$
8 **distribute securely** $B_i(\mathcal{P}, S_i) \rightarrow u_{i*}$
9 DistributeSecret(C_i)
10 **fVector()**

C. KEY DERIVATION PHASE

The key derivation process, which involves a concise set of linear operations, is described in detail in Algorithm 2 (and see Algorithm 3) and is illustrated in Figure 3. Initially, a set of vectors B_i is transformed into an orthonormal set $B_i^\perp$ using the Gram-Schmidt procedure. Subsequently, the first public vector f_1 is projected onto the subspace W_i, yielding a resultant vector $f^*_{(1,i)}$. The key derivation is completed by computing the inner product of $f^*_{(1,i)}$ with the second public vector f_2, thus producing the unique key K_i. The key K_i is not a vector in the subspace W_i; rather, it is a scalar value derived without revealing any properties of W_i, which we aim to keep secret. This ensures that the key itself does not disclose any information about the subspace W_i.

Any user $u_{ix} \in C_i$ can execute Algorithm 2 (and Algorithm 3) and effectively derive K_i, see Algorithm 2 (stat. 3,4,5,12). Besides, if $C_j \in Desc(C_i)$ then u_{ix} can also derive K_j, see Algorithm 2 (stat. 7-10). If $C_j \notin Desc(C_i)$ then u_{ix}

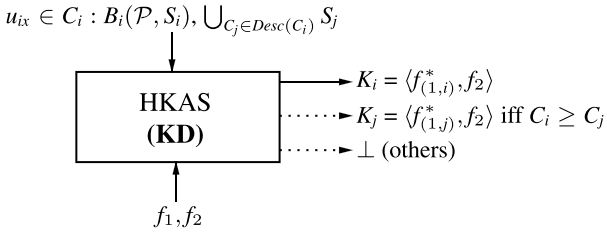

FIGURE 3. Illustration of algorithm 2, KD: key derivation.

unable to derive K_j due to the lack of S_j, see Algorithm 2 (stat. 11). Since $\mathcal{V}$ contains infinite subset of length $|S_j|$(or s), it is computationally impossible to obtain S_j. This is where the security of the key assignment scheme we propose is based.

Algorithm 2 Key Derivation Process

Input:
a user $u_{ix} \in C_i : B_i(\mathcal{P}, S_i), \bigcup_{C_j \in Desc(C_i)} S_j$
public information pub= (f_1, f_2)
Key Derivation:
$B_i^{\perp} \leftarrow$ [Algorithm 3].GramSchmidt(B_i)
$f^*_{(1,i)} \leftarrow$ [Algorithm 3].Projection($f_1, B_i^{\perp}$)
$K_i \leftarrow \langle f^*_{(1,i)}, f_2 \rangle$
if $C_j \in Desc(C_i)$ **then**
  $B_j^{\perp} \leftarrow$ [Algorithm 3].GramSchmidt($\mathcal{P} \cup S_j$)
  $f^*_{(1,j)} \leftarrow$ [Algorithm 3].Projection($f_1, B_j^{\perp}$)
  $K_j \leftarrow \langle f^*_{(1,j)}, f_2 \rangle$
  return (K_i, K_j)
else
  return $(K_i, \perp)$
return K_i

Algorithm 3 GramSchmidt and Projection Procedures

: **Proc GramSchmidt**(B_*)**:**
  $\#B_* = \{w_1, w_2, \ldots, w_n\}$
  $w'_1 = w_1$; $w''_1 = \frac{w'_1}{\|w'_1\|}$
  for $i = 2$ *to* n **do**
    $w'_i = w_i - \sum_{j=1}^{i-1} \frac{\langle w_i, w'_j \rangle}{\langle w'_j, w'_j \rangle} w'_j$
    $w''_i = \frac{w'_i}{\|w'_i\|}$
  return $B_*^{\perp} = \{w''_1, w''_2, \ldots, w''_n\}$
: **Proc Projection**($f_1, B_*^{\perp}$)**:**
  return $\sum_{i=1}^{n} \langle f_1, w''_i \rangle w''_i$

D. DYNAMIC UPDATE PHASE

In this section, we discuss the properties of dynamic key management, including key revocation, key rollover, adding/deleting new/existing classes, and users in the hierarchical structure. The most important feature of our scheme is that it performs all dynamic update operations, which is presented in Algorithm 4, with minimum computational complexity and space requirement by providing forward and backward secrecy properties. It is important to note that all the procedures in Algorithm 4 can only be executed by u_{11}.

The ClassReInitialize procedure (in Algorithm 4), which comprises a call statement 3.1 and a compound statement 3.2, is defined to reinitialize an existing class (calculating new S_i, B_i for C_i). Statement 3.1 initiates the revocation procedure for K_i of class C_i. With statement 3.2, S_i, B_i are created for the class C_i to be reinitialized, and W_i is constructed. B_i is then securely distributed to u_{i*}.

1) KEY REVOCATION

Key revocation is the process of invalidating or deactivating a cryptographic key (or key components) and is a crucial component of a hierarchical key assignment scheme. A class's secret key may need to be revoked for a number of reasons: compromise, end of lifecycle, policy changes, change in access control, or other reasons.

In our proposed scheme, Algorithm 4 includes the KeyRevocate procedure for the key revocation process. The discussion will not focus on the method of revocating the class key (see Algorithm 4 stat. 1.1), as it is directly related to the implementation of the scheme.

2) KEY ROLLOVER

The process of generating and using a new key instead of an existing one is known as key rollover (or rotation). A lightweight key rollover process is required to maintain the security and integrity of a hierarchical key assignment scheme. The KeyRollover procedure is part of Algorithm 4 with two statements (2.1, 2.2), which are call statements that execute the ClassReInitialize and [Algorithm 1]. DistributeSecret procedures, respectively, for C_i. With statement 2.2, the S_i is distributed securely to the users of all classes in $Anc(C_i)$.

Operations on classes or users and key rollover do not require modification of the public information pub=(f_1, f_2). If the pub must be modified for some reason, the [Algorithm 1].fvector() procedure must be executed. It should be noted that in this scenario, all K_i for $i : 1, \cdots, c$ will be changed.

3) ADDING A NEW CLASS OR USER

The ClassInsertion and UserInsertion procedures, respectively, in Algorithm 4 allow the addition of a new class or user to the hierarchical structure. The ClassInsertion procedure consists of 3 statements (4.1, 4.2, 4.3). With compound statement 4.1, S_i, B_i are created for the class C_i to be newly created, and W_i is constructed. B_i is then securely distributed to u_{i*}. With statement 4.2, S_i is distributed securely to users of all classes in $Anc(C_i)$. Finally, with statement 4.3, S_j sets of all classes in $Desc(C_i)$ are obtained and distributed securely to u_{i*}.

Let us consider Figure 1 and add the new class C_i to the hierarchy as follows.

$$\text{[Algorithm 4].ClassInsertion}(C_i, \overbrace{\{C_1\}}^{Anc(C_i)}, \overbrace{\{C_2, C_4, C_5\}}^{Desc(C_i)})$$

The UserInsertion procedure consists of two compound statements (6.1, 6.2). With statement 6.1, S_i is distributed securely to the user u_{ix} to be added to the C_i class. In the following for statement (6.2), S_j of each $C_j \in Desc(C_i)$ is obtained and distributed to u_{ix} securely.

4) DELETING AN EXISTING CLASS OR USER

Similarly, Algorithm 4 includes the ClassDeletion and UserDeletion procedures, respectively, to delete an existing class or user from the hierarchy. There are recursive call statements in both procedures (see stat. 5.2, 5.3, 7.3, 7.4). For example, the 5.2 recursive call statement means that the ClassReInitialize procedure will be executed for each $C_j \in Desc(C_i)$. On the other hand, ClassInsertion and ClassDeletion procedures require updating the access graph G. For readability and simplicity, the update of the access graph G is not included in the relevant procedures.

Algorithm 4 Dynamic Update Procedures

1 : **Proc KeyRevocate**(C_i)**:**
  1.1 **revocate** K_i
2 : **Proc KeyRollover**(C_i)**:**
  2.1 ClassReInitialize(C_i)
  2.2 [Algorithm 1].DistributeSecret(C_i)
3 : **Proc ClassReInitialize**(C_i)**:**
  3.1 KeyRevocate(C_i)
  3.2 Algorithm 1 stat. : 5-8
4 : **Proc ClassInsertion**(C_i, $Anc(C_i)$, $Desc(C_i)$)**:**
  4.1 Algorithm 1 stat. : 5-8
  4.2 [Algorithm 1].DistributeSecret(C_i)
  4.3 **for** C_j *in* $Desc(C_i)$ **do**
    distribute securely $\mathcal{S}^{u_{11}}[C_j] \rightarrow u_{i*}$
5 : **Proc ClassDeletion**(C_i)**:**
  5.1 KeyRevocate(C_i)
  5.2 ClassReInitialize*($Desc(C_i)$)
  5.3 [Algorithm 1].DistributeSecret*($Desc(C_i)$)
6 : **Proc UserInsertion**(u_{ix}, C_i)**:**
  6.1 **distribute securely** $\mathcal{S}^{u_{11}}[C_i] \rightarrow u_{ix}$
  6.2 **for** C_j *in* $Desc(C_i)$ **do**
    distribute securely $\mathcal{S}^{u_{11}}[C_j] \rightarrow u_{ix}$
7 : **Proc UserDeletion**(u_{ix}, C_i)**:**
  7.1 ClassReInitialize(C_i)
  7.2 [Algorithm 1].DistributeSecret(C_i)
  7.3 ClassReInitialize*($Desc(C_i)$)
  7.4 [Algorithm 1].DistributeSecret*($Desc(C_i)$)

V. EXPERIMENTAL RESULTS

We implemented the proposed hierarchical key assignment scheme on a computer running Macintosh OS Ventura, equipped with a 3.2 GHz 6-Core Intel Core i7 processor and 16 GB of 2667 MHz DDR4 memory. The implementation was done using the C++ programming language in the Xcode IDE (version 14.3). Below is a brief description of the algorithms.

Algorithm 1. The system-preparation and distribution processes. The inputs of Algorithm 1 (in the implementation) are selected as ($\mathcal{V} = \mathbb{R}^m, n, s = 1$) with $m > n$ and the dot product is used as the inner product. Algorithm 2. Key derivation process, Algorithm 3. Gram-Schmidt and Projection procedures, Algorithm 4. Dynamic update process.

In this section, the results of the performance and security analysis and comparison with other schemes are given. The experimental results show that the proposed scheme is scalable, flexible, and cost-effective in terms of key-derivation and dynamic update phases.

A. PERFORMANCE ANALYSIS

As with many other cryptographic schemes, the efficiency of a hierarchical key assignment scheme is evaluated based on the public-private storage need and the computational complexity of the key derivation and key rollover processes.

In our scheme, all vectors, including (f_1, f_2), are all elements of $\mathcal{V}$ (equivalent to $\mathbb{R}^m$) and have m components, namely $\forall w_*, f_1, f_2 \in \mathbb{R}^m$. To comprehensively convey the requirements for both public and private storage and to facilitate precise comparisons with other schemes, let's assume that all vectors in the scheme are represented by ρ (security parameter, see Definition 2) bits, just like the secret keys of classes.

1) PUBLIC STORAGE

There are two public keys $f_1, f_2 \xleftarrow{\$} \mathcal{V}, (f_1, f_2 \notin W_1, \cdots, W_c)$ for the entire scheme (see [Algorithm 1].fvector procedure). Therefore, the public storage need of our scheme is 2ρ bits.

2) PRIVATE STORAGE

Each user $u_{ix} \in C_i$ must keep the basis B_i and $\bigcup_{C_j \in Desc(C_i)} S_j$ as private. Let's assume that $|B_i| = n, |S_i| = s, |\mathcal{P}| = n - s, |Desc(C_i)| = \alpha$, the number of vectors to be stored secretly/privately by each u_{ix} is $n+\alpha*s$. If the default value of 1 is used for the parameter s, the result is the need for private storage $(n + \alpha)$. As a result, the bit length of the private data that each user should keep secret is $(n + \alpha)\rho$.

3) KEY DERIVATION TIME COMPLEXITY

Since $\forall B_i \in \mathcal{B}, |B_i| = n$, the cost of key derivation will be the same for all users, regardless of the class of a user within the hierarchy. In other words, the computational cost of the key derivation depends on the n (and also m) rather than the hierarchy's number of classes (namely c) or the security level (depth in G) of the classes.

The key derivation process, Algorithm 2, requires Gram-Schmidt orthogonalization ([Algorithm 3].Gram-Schmidt), orthogonal projection ([Algorithm 3].Projection)

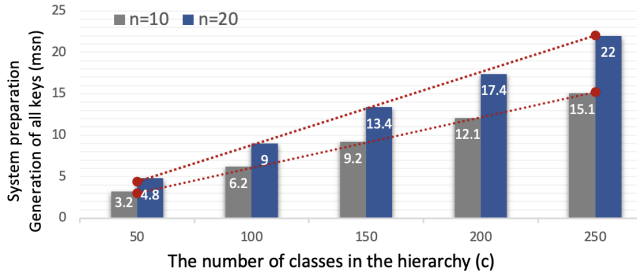

FIGURE 4. System-preparation time cost consumed by u_{11}, default value(s): $m = 280$ ($\mathcal{V} = \mathbb{R}^{280}$) and $s = 1$.

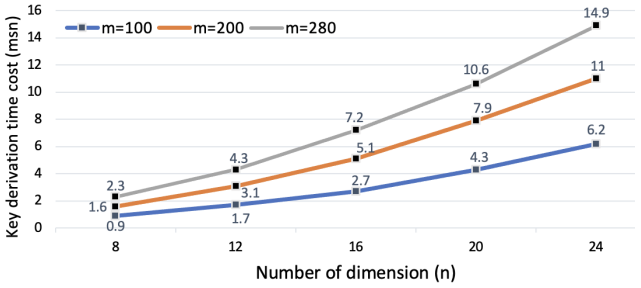

FIGURE 5. Key derivation time cost, consumed by u_{ix}, default value(s): $s = 1$.

procedures, and ultimately, the computation of the inner product. The sum of the complexity of these three steps creates the time complexity of the key derivation for any C_i.

Let's denote the total number of multiplications with M, the number of divisions with D and the number of inner products with I.

$$M : \sum_{i=2}^{n}(i-1)+n = \frac{n(n+1)}{2} \text{ multiplications,}$$

$$D : 1+\sum_{i=2}^{n} i = \frac{n(n+1)}{2} \text{ divisions,}$$

$$I : 1+\sum_{i=2}^{n}(2(i-1)+1)+n+1 = n^2+n+1$$

inner products.

In conclusion, for a user $u_{ix} \in C_i$, key derivation complexity of one of K_i or K_j, iff $C_j \in Desc(C_i)$, is $\mathcal{O}(n^2)$.

The results of the implementation (or simulation) were derived from the scenarios shown in Figures 4 and 5. The first scenario (Figure 4) illustrates the relationship between the number of classes (c), the length of the basis sets (n), and the time (measured in milliseconds) required for system preparation—referred to as the generation of all keys. The time necessary for the distribution process, as described in Algorithm 1 (stat. 8,9), was not studied. This is because it largely depends on the specific implementation of the scheme, especially in terms of distribution methods.

Figure 5 (the second scenario) shows how the length of the basis sets (n) and (m) such that $\mathcal{V} = \mathbb{R}^m$ affects the cost of the key derivation procedure. The cost of key derivation is a crucial consideration for HKA schemes. When examining the simulation results, we conclude that our proposed scheme can be applied effectively in real-time HKA applications, including cloud computing, secure communication in IoT, healthcare data protection, and others, with regard to system preparation and key derivation time costs (and also space requirements).

4) KEY ROLLOVER TIME COMPLEXITY

The time complexity of the key rollover process (procedure) is approximately equal to the complexity of the [Algortihm 1].DistributeSecret procedure, see Algorithm 4 stat. 2.2. The complexity required to change the K_i for a given C_i is $\mathcal{O}(|Anc(C_i)|)$.

B. SECURITY NOTIONS AND ANALYSIS

In this section, we show through formal and informal security analysis that the proposed scheme is secure against a variety of attacks [28], [34], [35]. Several of the initial schemes did not have a formal security analysis. However, this deficiency has been progressively resolved, starting with the research conducted by Atallah et al. [26]. They introduced two distinct security notions: security against key recovery (KR-security) and security in terms of key indistinguishability (KI-security). The strengthened version of KI-security, termed strong key indistinguishability security (S-KI-security), was proposed by Freire et al. [34].

1) KR-security

It is computationally impossible for one or more users, namely a coalition of users, from the set

$$F_{C_i} = \cup_{u_* \notin C_i \wedge u_* \notin Anc(C_i)} u_*$$

to derive K_i. Since Algorithm 1 provides precise and rigorous definitions $S_i \neq S_j$, $S_i \cap S_j = \{\}$ iff $i \neq j$ and $\mathcal{V}$ contains infinite subset of length $|S_i|$(or s), any subset of F_{C_i} cannot derive (or recover) K_i.

Assuming that users in F_{C_i} combined their privates S_*, since $S_i \not\subset \cup_{u_* \in F_{C_i}} S_*$ and all vectors in the set $\mathcal{S}$ are linearly independent, it is computationally infeasible to get (or recover) S_i (and also K_i, see Algorithm 2) from given (or combined) information. Therefore, our scheme is secure against key recovery (collusive) attacks.

2) KI-security

In this security model, the adversary lacks knowledge of the secret key K_i and cannot differentiate K_i from a random string of equivalent length. In our scheme, for each class C_i, the secret key K_i is represented as a number without any additional patterns or properties, providing KI-security. It is important to note that KI-security inherently implies KR-security, meaning that if a system is KI-secure, it is also KR-secure.

3) RESISTANCE TO PRIVILEGE CREEP PROBLEM

In our scheme, for each class $C_i \in \mathcal{C}$, B_i is stored secretly by users u_{i*} and S_i is stored secretly by $\bigcup_{u_* \in Anc(C_i)} u_*$, not stored elsewhere. The dynamic update procedures KeyRollover, ClassReInitialize, ClassDeletion, and UserDeletion in Algorithm 4 call the KeyRevocate procedure directly or indirectly. Since it is not possible for a revoked key K_i or key component S_i to be in use or reused for another class, our scheme is secure against the problem of resistance to privilege creep.

4) S-KI-security

We perform a formal security analysis, supported by a security proof, with the S-KI-security model that implies both KI-security and KR-security. Since the S-KI-security model is polynomially equivalent for both static and dynamic (adaptive) adversaries, we examine the security model with respect to a static adversary $\mathcal{A}_{stat}$ [26], [28], [34], [35].

For a given access graph $G = (\mathcal{C}, E)$, consider a static adversary A_{stat}, who wants to attack a class $C_i \in \mathcal{C}$. Using the *Gen* algorithm on the access graph G, the security experiment generates (B_*, K_*, pub). The adversary is then given two functions $Corrupt_{C_i}(B_*)$ and $Key_{G,C_i}(K_*)$ as follows.

$$Corrupt_{C_i}(B_*) = \{B_j \in \mathcal{B} | (C_j \neq C_i) \wedge (C_j \notin Anc(C_i))\}$$
$$Key_{G,C_i}(K_*) = \{K_j | C_j \in Anc(C_i)\}$$

A_{stat} can corrupt all users in F_{C_i} by $Corrupt_{C_i}(B_*)$ function and gain access to the keys associated with ancestor classes by using $Key_{G,C_i}(K_*)$. During the challenge phase, A_{stat} is provided with the secret key K_i or a random string of the same length $\{0,1\}^\rho$, with the purpose of distinguishing between the two cases. The following is the formal definition of S-KI-security.

Definition 3 ([S-KI-ST]): Let Γ be a set of access graphs corresponding to partially ordered sets (hierarchies), let $G = (\mathcal{C}, E) \in \Gamma$, let (Gen, Der) be a HKAS for Γ, and let A_{stat} be a static adversary which attacks a class C_i. Consider the following experiment:

Experiment $\mathbf{Exp}^{S-KI-ST}_{G,A_{stat}}(1^\rho)$
$C_i \leftarrow A_{stat}(1^\rho, G)$
$(B_*, K_*, pub) \leftarrow Gen(1^\rho, G)$
$\beta \xleftarrow{r} \{0,1\}$
If $\beta = 1$ *then* $T \leftarrow K_i$ *else* $T \xleftarrow{r} \{0,1\}^\rho$
$d \leftarrow A_{stat}(1^\rho, G, pub, Corrupt_{C_i}(B_*), Key_{G,C_i}(K_*), T)$
return d

For a given $G \in \Gamma$, the advantage of A_{stat} in the experiment can be defined as:

$$Adv^{S-KI-ST}_{G,A_{stat}}(1^\rho) = 2|Pr[Exp^{S-KI-ST}_{G,A_{stat}}(1^\rho) = \beta] - 1/2|$$

[34] If we symbolize $\mathbf{Exp}^{S-KI-ST,\lambda}_{G,A_{stat}}(1^\rho)$ with $\lambda \in \{0,1\}$, $\beta = \lambda$ we get

$$Adv^{S-KI-ST}_{G,A_{stat}}(1^\rho) = \Big| Pr[Exp^{S-KI-ST,1}_{G,A_{stat}}(1^\rho) = 1] - Pr[Exp^{S-KI-ST,0}_{G,A_{stat}}(1^\rho) = 1] \Big|$$

The hierarchical key assignment scheme is said to be secure in the sense of strong key indistinguishability (S-KI-ST) with respect to a static adversary A_{stat}, if $Adv^{S-KI-ST}_{G,A_{stat}}(1^\rho)$ is negligible for each $G \in \Gamma$ and $C_i \in G$. Namely, it is impossible for any A_{stat} to do better than succeeding with probability 1/2 by outputting a random guess for S-KI-ST.

In conclusion, clearly, A_{stat} with functions $Corrupt_{C_i}(B_*)$ and $Key_{G,C_i}(K_*)$ cannot distinguish a secret key K_i from any K_j iff $i \neq j$ (or generally $\{0,1\}^\rho$), so the scheme provides S-KI-security.

Finally, let us now discuss the security of our proposed scheme with respect to the difficulty of the CVP-IPS. Recall that we are dealing with an adversary A_{stat} whose objective is to compromise a specific class C_i. If A_{stat} is outside the hierarchy, A_{stat} has no access to any vector of the basis B_i. However, if A_{stat} is inside the hierarchy but is not a member of the classes $\{C_i, Anc(C_i)\}$, A_{stat} knows the subset $\mathcal{P}$. To explore a more generalized form of the second scenario, let's delve into the following selective security game.

[Setup]

- Let $\mathcal{V}$ be an infinite-dimensional inner product space, and let W_i be a subspace of $\mathcal{V}$ with $\dim(W_i) = n$.
- The challenger (u_{11} by default) runs the $\mathbf{Gen}(\mathbf{1}^\rho, \mathbf{G})$ algorithm for C_i and generates (B_i, K_i, pub). The challenger then randomly chooses r vectors $\{w_1, w_2, \cdots, w_r\}$, where $r \leq n-1$, from B_i and sends them to A_{stat} with $(\mathcal{V}, n)$.

[Challenge]

- The A_{stat}'s challenge is to select $n-r$ additional vectors $\{w'_{r+1}, w'_{r+2}, \cdots, w'_n\}$ from $\mathcal{V}$ to complete B_i.

[Adversary's Goal]

- A_{stat} succeeds in the game if A_{stat} can correctly choose the remaining $n-r$ vectors such that $\{w_1, w_2, \cdots, w_r, w'_{r+1}, w'_{r+2}, \cdots, w'_n\}$ forms a complete, linearly independent basis for W_i.

The game demonstrates the difficulty for an adversary A_{stat} in correctly guessing or computing the remaining vectors to complete the basis set B_i given the limited initial knowledge. Given the infinite-dimensional nature of $\mathcal{V}$, this probability is expected to be negligible.

$$P\left(\bigwedge_{j=r+1}^{n} \left(w'_j \in W_i \wedge w'_j \nparallel \{w_1, w_2, \ldots, w_r\}\right)\right) = negl(\eta)$$

where $\bigwedge$ is the logical AND operator, used here to indicate that all conditions in the sequence must be satisfied. $w'_j \nparallel \{w_1, w_2, \ldots, w_r\}$ denotes that the vector w'_j is linearly independent of the set of vectors, and $negl(\eta)$ represents a negligible function in terms of the security parameter η.

C. COMPARISON WITH OTHER SCHEMES

Table 1 compares our scheme with other well-known HKA schemes in the literature. In our detailed comparison table, public storage (information) is measured for the entire access scheme ([22], [26], [29], [30], [31], [32], [33], [34], [35], [36],

[37], [39], [41], our scheme), private storage (information) is measured per class ([22], [26], [29], [30], [31], [32], [33], [34], [35], [36], [37], [41]), or per user (our scheme). The scheme proposed in [39] the requirement for private storage is stated for the entire scheme. In the comparison, various parameters are considered, and their respective notations are found at the bottom of Table 1.

In their papers [22] and [26], Atallah et al. presented two different HKA schemes. The first one is based on PRF, while the second one is based on a combination of PRF and CPA-secure symmetric encryption scheme. The first scheme provides KR-security, while the second one offers KI-security. Each class in both schemes has only one defined private key. In the first scheme, the amount of public information is directly related to the number of edges. In the second scheme, the amount of public information increases in proportion to the number of classes and edges in the access hierarchy. Both schemes utilize symmetric operations for key derivation, with the derivation cost increasing linearly with the number of levels between classes.

In [29], D'Arco et al. proposed KI-secure scheme based on the KR secure Akl and Taylor scheme [2] and the Goldreich and Levin hard-core bit (GL bit) [5] with RSA security assertion. Given that the key derivation and public storage (roughly) are determined by ℓ, their design is suboptimal.

In [30], Lin et al. proposed a secure HKAS based on elliptic curve cryptosystems. From a complexity standpoint, their scheme is inefficient due to large public storage requirements which depend on multiple parameters (c, k_i, ρ), and key derivation complexity determined by the time-consuming subprocess.

In [31], De Santis et al. introduced methods rooted in CPA-secure symmetric encryption and public-key broadcast encryption. The former assigns one private key to each class, and the public storage demand increases proportionally to the number of classes and edges in the graph. The key derivation time depends roughly on h symmetric decryption. As the directed path between classes lengthens, key derivation costs rise.

Ateniese et al. proposed two time-bound key assignment schemes that achieve KI-security [33]. The first method, Two-Level Encryption-Based Construction (TLEBC), relies on a symmetric encryption scheme and operates under the IND-P1-C0 security assumption. The second method, Two-Level Pairing-Based Construction (TLPBC), is based on bilinear maps with the BDDH security assumption. Unlike the previous schemes that we have analyzed, the key derivation process involves only one decryption in the first scheme and one pairing evaluation in the second scheme, making it highly efficient. The key derivation does not depend on the number of classes in the hierarchy, the number of edges, or the distance between the classes. However, both schemes have a significant drawback, which is the very large amount of public storage requirement. Besides, another disadvantage of the second scheme is that the private storage requirement is equal to the number of distinct time periods.

In [32], a key assignment scheme for arbitrary posets was introduced, offering KI-security based on the computational hardness of factoring. The storage needed for private data is determined by the poset with w. Since key derivation requires roughly $\ell.h$ modular squaring, especially when h is large, the derivation process is inefficient. In [34] Freire et al. proposed two different constructions that provide S-KI-security based on pseudorandom function (PRF) and forward secure pseudorandom generator (FS-PRG), respectively. In both constructions, the public storage requirements are zero, and the private storage requirements are determined by the poset with w. The key derivation process in both schemes is inefficient because it requires roughly h multiplications, where h is the length of the directed path between classes (or the number of levels between classes).

In [35], Tang et al. proposed a directed HKA scheme based on linear geometry and provides S-KI-security with respect to PRF security assumption. Their scheme requires lightweight computation over a finite field and provides dynamic update processes. The presented scheme, although effective, has the downside of requiring more public storage space compared to other HKA schemes. According to [35], their scheme has a well-optimized balance between computational requirements and storage space utilization. The costs of key derivation for the class itself and its descendants are $2M + A$ and $4M + 2A$, respectively, which is effective.

In [36], Castiglione et al. proposed two constructions, called shared encryption based construction (SEBC), and threshold broadcast encryption based construction (TBEBC), respectively. While both constructions utilize one private key for each class in the hierarchy, the public storage requirement of the second method is lower than that of the first method. In [37], only one symmetric decryption is required to derive the key, which is more efficient than the constructions mentioned above in [36]. Reference [37] requires one key for private storage, public storage is bounded by $|E_{TL}|$.

In [39], Celiktas et al. proposed a hierarchical key management protocol based on threshold (approval/confirmation) for key access in cloud computing. This scheme uses secret-splitting, shamir's(t, m) threshold algorithm for its cryptographic construction and provides KI-security under the *TSBITS* security assumption. In their protocol, a user can access the secret key on both internal and external networks after receiving approval. That is, a user can only access cloud data with sufficient approvals. The scheme uses the topological ordering of a directed graph including self-loop. That is, there can only be one group (class) at each security level, in other words, the scheme does not support horizontal scaling, which is the most important shortcoming of the scheme. Additionally, vertical scaling (security level) can be costly.

In [41], Celiktas et al. presented a hierarchical key assignment scheme based on inner product spaces. They utilized orthogonal projection (OP) as a mathematical tool. Their scheme is the first work to adapt the orthogonal projection method to HKA schemes and provides

TABLE 1. Comparison with other schemes.

Schemes	Public Storage	Private Storage	Key Derivation	Type of Security	Security Assumption
Atallah et al. [22]	$\rho\|E\|$	ρ	$(T_H + T_{xor})h$	KR	PRF
Atallah et al. [26]	$\rho\|E\|+\rho_1 c$	ρ	$(T_{sym\text{-}dec} + T_{PRF})h$	KI	PRF+CPA-secure
D'Arco et al. [29]	$\rho 2(c(1+\ell)+2)$	ρ_2	$T_{exp\text{-}mod}\,\ell$	KI	RSA (random exp.)
Lin et al. [30]	$(3c + \sum_{i=2}^{c} k_i + 4)\rho$	ρ	$T_{ecc\text{-}dec} + T_{pol} + 2T_H$	N/A	OWHF+CPA-secure
De Santis et al. [31]	$(\|E\| + 2c)\rho_1$	ρ	$(h+2)T_{sym\text{-}dec}$	KI	CPA-secure
Ateniese et al. TLEBC [33]	max:c^2t^3	One	$T_{sym\text{-}dec}$	KI	IND-P1-C0
Ateniese et al. TLPBC [33]	max:c^2	max:t	T_P	KI	BDDH
Freire et al. [32]	ρ_2	$w.\rho_2$	$\ell(h+1)T_{sqr\text{-}mod}$	KI	BBS
Freire et al. PRF-based [34]	Zero	$w.\rho$	$(h+1)T_{PRF}$	S-KI	PRF
Freire et al. FS-PRG-based [34]	Zero	$w.\rho_2$	$(h+1)T_{FS\text{-}PRG}$	S-KI	FS-PRG
Tang et al. [35]	$\rho(c^2+1)$	4ρ	$4M + 2A$	S-KI	PRF
Castiglione et al. SEBC [36]	max:$(\|E\| + c)\sigma_1\rho$	ρ	$\sigma_2 T_{sym\text{-}dec} + T_{rec}$	KI	IND-P1-C0+PSSS
Castiglione et al. TBEBC [36]	$(1+z)c$	ρ	$\sigma_3 T_{sym\text{-}dec}$	KI	TBEBC
Castiglione et al. TLEBC [37]	$\|E_{TL}\|$	ρ	$T_{sym\text{-}dec}$	KI	IND-P1-C0+IND-D-A
Celiktas et al. [39]	Zero	min: One, max: c	$\mathcal{O}(d^2.\log p)$	KI	TSBITS
Celiktas et al. [41]	One	min: One, max: b	Op_t	S-KI	LICV
Our Scheme	2ρ	$(n+\alpha)\rho$	$T_{GS} + T_{OP}$	S-KI	CVP-IPS

Notation:
ρ : security parameter, bit size of key;
ρ_1 : the size of the ciphertext in a semantically secure symmetric key encryption scheme;
ρ_2 : the security parameter for a pseudorandom number generator;
ℓ : the bit size of the keys used in the scheme [29];
$|E|$: the number of edges in the access graph G;
c : $|\mathcal{C}|$, the number of classes in the access graph G;
n: $|B_i|$, the length of basis sets;
α: the number of classes in the $Desc(C_i)$;
h: the length of the directed path from C_i to C_j ($C_i \geq C_j$);
t: number of distinct time periods;
d: the degree of the interpolation polynomial;
p: a prime number is utilized to form a prime field F_p;
w: width of the poset;
z: the number of sets that are qualified to access the current class;
b: the maximum dimension of a subspace associated with any class at the bottom in the hierarchy [41];
k_i: the degree of the polynomial $f(x)$ [30];
σ_1: a constant based on the symmetric encryption method used;
σ_2: number of classes corresponding to each layer to compute the encryption key;
σ_3: number of classes in a given qualified set for a current class;
T_H: the time required for the computation of a hash function;
T_P: the time needed for one pairing evaluation;
T_{GS}: the time required for the computation of gram schmidt orthogonalization procedure (see Algorithm 3);
T_{OP}: the time required for the computation of orthogonal projection (see Algorithm 3);
T_{pol}: the time required to calculate the value of a polynomial;
T_{rec}: the time required to use the recovery algorithm of the perfect secret sharing scheme;
T_{xor} : the amount of time required to perform XOR operations;
T_{PRF}: the time required to compute the output of a PRF;
$T_{FS\text{-}PRG}$: the time required to compute the output of an FS-PRG;
$T_{sym\text{-}dec}$: the time required for decryption in a symmetric key encryption scheme;
$T_{ecc\text{-}dec}$: the time required for decryption in an elliptic curve-based (ECC) public key encryption scheme;
$T_{exp\text{-}mod}$: the time required for the modular exponentiation;
$T_{sqr\text{-}mod}$: the time required for modular squaring;
M, A: computational cost of multiplication and addition over F_q, respectively;
OP_t : orthogonal projection time cost [41];
$|E_{TL}|$: the number of edges of two-level graph $G_{TL} = (V_{TL}, E_{TL})$ [37];
PRF: pseudorandom function;
BBS: blum blum shub, pseudorandom number generator;
N/A : Not applicable
LICV : linearly independent chosen vectors;
OWHF: one way hash function;
PSSS: perfect secret sharing schemes;
BDDH: bilinear decisional Diffie-Hellman;
SEBC: shared encryption based construction;
TLEBC: two-level encryption-based construction;
TLPBC: two-level pairing-based construction;
TBEBC: threshold broadcast encryption-based construction;
TSBITS : threshold scheme based information-theoretic secure;
FS-PRG: forward secure pseudorandom generator;
CVP-IPS: closest vector problem (CVP) in an inner product space (IPS) ;
CPA-secure: chosen plaintext attack secure;
IND-P1-C0: non-adaptive chosen plaintext attack [23];
IND-D-A (IND-DYN-AD): indistinguishability security model (dynamic, adaptive);

S-KI-security under the LICV security assumption. The public storage requirement is only one, and the maximum private storage requirement per class is b. Typically, the cost of key derivation increases with the length of the basis set (as well as with the properties of the inner product space).

Reference [41] employs a left-right, top-down approach in which all vectors used in the hierarchy are included in the basis set defined for the root class. In other words, the basis sets for high security level classes are longer. The key derivation for classes at higher levels in the hierarchy is less efficient and takes more time compared to classes at lower security levels. Additionally, defining different basis sets for classes naturally increases the basis set lengths of higher classes.

Our proposal introduces a direct key assignment scheme, which is supported by rigorous mathematical definitions and algorithmic syntax. Our scheme provides S-KI-security under the CVP-IPS security assumption. The public storage required for the entire scheme is two, while the private storage required for each user increases directly proportional to $(n + \alpha)$. The complexity of the key derivation is $\mathcal{O}(n^2)$ and depends strongly on the Gram-Schmidt orthogonalization algorithm (see ([Algorithm 3].GramSchmidt procedure). Since the length of the basis set defined for each class in the hierarchy is the same, the key derivation cost will be equal for each user in the access hierarchy. In our scheme, unlike [41] and many other schemes, there may be more than one directed path between a class at the higher level and a class at the lower level. With these and other features, the scheme we propose fully meets all dynamic processes (see Algorithm 4).

Unlike many schemes proposed in the literature, some abstractions have been made in the scheme we propose. These are how the private key components will be distributed, where they will be stored and, accordingly, how the key revocation process will be carried out. For our proposed scheme to be adaptable, we assume that private key components will be stored by users of the hierarchy. In practice, different distribution/storage and key revocation scenarios can be applied for users belonging to a class in the upper or lower security level. For example, any protocol implementing our scheme can employ a key distribution center (KDC) in the hierarchy under the control of the data owner. In this case, the private key components are not distributed to the users, but are stored in the KDC by the data owner. Users in the hierarchy can securely access the KDC and obtain private key components.

VI. CONCLUSION

This study introduces an innovative direct hierarchical key assignment scheme (HKAS), utilizing CVP-IPS as its core mathematical tool. The scheme is notable for its scalability, flexibility, cost-effectiveness, and high performance. One key advantage is that the lengths of the basis sets are the same across all classes, ensuring identical key derivation costs for users of any class. It also boasts a robust mathematical framework, which is particularly efficient in handling dynamic update operations. Departing from traditional top-down structures, this HKAS achieves both vertical and horizontal scalability and is distinguished by its strong key indistinguishability security (S-KI-security).

REFERENCES

[1] P. V. E. Boas, "Another NP-complete problem and the complexity of computing short vectors in a lattice," Dept. Math., Univ. Amsterdam, Tech. Rep. 81-04, 1981.

[2] S. G. Akl and P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," *ACM Trans. Comput. Syst.*, vol. 1, no. 3, pp. 239–248, Aug. 1983.

[3] S. MacKinnon, P. Taylor, H. Meijer, and S. Akl, "An optimal algorithm for assigning cryptographic keys to control access in a hierarchy," *IEEE Trans. Comput.*, vol. 34, no. 9, pp. 797–802, 1985.

[4] R. Sandhu, "Cryptographic implementation of a tree hierarchy for access control," *Inf. Process. Lett.*, vol. 27, no. 2, pp. 95–98, 1988.

[5] O. Goldreich and L. A. Levin, "A hard-core predicate for all one-way functions," in *Proc. ACM STOC*, 1989, pp. 25–32.

[6] L. Harn and H.-Y. Lin, "A cryptographic key generation scheme for multilevel data security," *Comput. Secur.*, vol. 9, no. 6, pp. 539–546, Oct. 1990.

[7] C.-C. Chang, R.-J. Hwang, and T.-C. Wu, "Cryptographic key assignment scheme for access control in a hierarchy," *Inf. Syst.*, vol. 17, no. 3, pp. 243–247, May 1992.

[8] H. T. Liaw, S. J. Wang, and C. L. Lei, "A dynamic cryptographic key assignment scheme in a tree structure," *Comput. Math. Appl.*, vol. 25, no. 6, pp. 109–114, Mar. 1993.

[9] M.-S. Hwang, W.-P. Yang, and C.-C. Chang, "Modified Chang–Hwang–Wu access control scheme," *Electron. Lett.*, vol. 29, no. 24, pp. 2095–2096, 1993.

[10] H.-T. Liaw and C.-L. Lei, "An optimal algorithm to assign cryptographic keys in a tree structure for access control," *BIT*, vol. 33, no. 1, pp. 46–56, Mar. 1993.

[11] H. Min-Shiang, "A cryptographic key assignment scheme in a hierarchy for access control," *Math. Comput. Model.*, vol. 26, no. 2, pp. 27–31, Jul. 1997.

[12] C.-H. Lin, "Hierarchical key assignment without public-key cryptography," *Comput. Secur.*, vol. 20, no. 7, pp. 612–619, Oct. 2001.

[13] W.-G. Tzeng, "A time-bound cryptographic key assignment scheme for access control in a hierarchy," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 1, pp. 182–188, Jan. 2002.

[14] V. R. L. Shen and T.-S. Chen, "A novel key management scheme based on discrete logarithms and polynomial interpolations," *Comput. Secur.*, vol. 21, no. 2, pp. 164–171, Mar. 2002.

[15] D. Micciancio and S. Goldwasser, *Complexity of Lattice Problems: A Cryptographic Perspective*. Berlin, Germany: Springer, 2002.

[16] X. Yi and Y. Ye, "Security of tzeng's time-bound key assignment scheme for access control in a hierarchy," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 4, pp. 1054–1055, Jul. 2003, doi: 10.1109/TKDE.2003.1209023.

[17] H. Y. Chien and J. K. Jan, "New hierarchical assignment without public key cryptography," *Comput. Secur.*, vol. 22, pp. 523–526, Sep. 2003.

[18] H.-Y. Chen, "Efficient time-bound hierarchical key assignment scheme," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 10, pp. 1301–1304, Oct. 2004, doi: 10.1109/TKDE.2004.59.

[19] T.-S. Chen, Y.-F. Chung, and C.-S. Tian, "A novel key management scheme for dynamic access control in a user hierarchy," in *Proc. 28th Annu. Int. Comput. Softw. Appl. Conf.*, 2004, pp. 396–401.

[20] X. Yi, "Security of Chien's efficient time-bound hierarchical key assignment scheme," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 9, pp. 1298–1299, Sep. 2005, doi: 10.1109/TKDE.2005.152.

[21] F.-G. Jeng and C.-M. Wang, "An efficient key-management scheme for hierarchical access control based on elliptic curve cryptosystem," *J. Syst. Softw.*, vol. 79, no. 8, pp. 1161–1167, Aug. 2006.

[22] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, "Dynamic and efficient key management for access hierarchies," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2006, pp. 190–202.

[23] J. Katz and M. Yung, "Characterization of security notions for probabilistic private-key encryption," *J. Cryptol.*, vol. 19, no. 1, pp. 67–95, Jan. 2006.

[24] E. Bertino, N. Shang, and S. S. Wagstaff, "An efficient time-bound hierarchical key management scheme for secure broadcasting," *IEEE Trans. Dependable Secure Comput.*, vol. 5, no. 2, pp. 65–70, Apr. 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," *Inf. Sci.*, vol. 178, no. 1, pp. 230–243, Jan. 2008.
[26] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, "Dynamic and efficient key management for access hierarchies," *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 3, pp. 1–43, Jan. 2009.
[27] H.-M. Sun, K.-H. Wang, and C.-M. Chen, "On the security of an efficient time-bound hierarchical key management scheme," *IEEE Trans. Dependable Secure Comput.*, vol. 6, no. 2, pp. 159–160, Apr. 2009.
[28] A. Castiglione, A. De Santis, and B. Masucci, "Key indistinguishability versus strong key indistinguishability for hierarchical key assignment schemes," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 4, pp. 451–460, Jul. 2016, doi: 10.1109/TDSC.2015.2413415.
[29] P. D'Arco, A. De Santis, A. L. Ferrara, and B. Masucci, "Variations on a theme by Akl and Taylor: Security and tradeoffs," *Theor. Comput. Sci.*, vol. 411, no. 1, pp. 213–227, Jan. 2010.
[30] Y.-L. Lin and C.-L. Hsu, "Secure key management scheme for dynamic hierarchical access control based on ECC," *J. Syst. Softw.*, vol. 84, no. 4, pp. 679–685, Apr. 2011.
[31] A. D. Santis, A. L. Ferrara, and B. Masucci, "Efficient provably-secure hierarchical key assignment schemes," *Theor. Comput. Sci.*, vol. 412, no. 41, pp. 5684–5699, Sep. 2011.
[32] E. Freire and K. Paterson, "Provably secure key assignment schemes from factoring," in *Information Security and Privacy* (Lecture Notes in Computer Science), vol. 6812, U. Parampalli and P. Hawkes, Eds. Berlin, Germany: Springer, 2011, pp. 292–309.
[33] G. Ateniese, A. De Santis, A. L. Ferrara, and B. Masucci, "Provably-secure time-bound hierarchical key assignment schemes," *J. Cryptol.*, vol. 25, no. 2, pp. 243–270, Apr. 2012, doi: 10.1007/s00145-010-9094-6.
[34] E. S. V. Freire, K. G. Paterson, and B. Poettering, "Simple, efficient and strongly KI-secure hierarchical key assignment schemes," in *Topics in Cryptology—CT-RSA* (Lecture Notes in Computer Science), vol. 7779, E. Dawson, Ed. Berlin, Germany: Springer, 2013, doi: 10.1007/978-3-642-36095-4_7.
[35] S. Tang, X. Li, X. Huang, Y. Xiang, and L. Xu, "Achieving simple, secure and efficient hierarchical access control in cloud computing," *IEEE Trans. Comput.*, vol. 65, no. 7, pp. 2325–2331, Jul. 2016, doi: 10.1109/TC.2015.2479609.
[36] A. Castiglione, A. De Santis, B. Masucci, F. Palmieri, A. Castiglione, J. Li, and X. Huang, "Hierarchical and shared access control," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 850–865, Apr. 2016, doi: 10.1109/TIFS.2015.2512533.
[37] A. Castiglione, A. De Santis, B. Masucci, F. Palmieri, A. Castiglione, and X. Huang, "Cryptographic hierarchical access control for dynamic structures," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 10, pp. 2349–2364, Oct. 2016, doi: 10.1109/TIFS.2016.2581147.
[38] B. Celiktas, İ. Çelikbilek, and E. Özdemir, "A higher level security protocol for cloud computing," in *Proc. 4th Int. Conf. Comput. Sci. Eng. (UBMK)*, Samsun, Turkey, Sep. 2019, pp. 97–101, doi: 10.1109/UBMK.2019.8907019.
[39] B. Celiktas, I. Celikbilek, and E. Ozdemir, "A higher-level security scheme for key access on cloud computing," *IEEE Access*, vol. 9, pp. 107347–107359, 2021.
[40] S. Guzey, G. Karabulut Kurt, and E. Ozdemir, "A group key establishment scheme," 2021, *arXiv:2109.15037*.
[41] B. Celiktas, S. Guzey, and E. Ozdemir, "An inner product space-based hierarchical key assignment scheme for access control," *TechRxiv*, 2022, doi: 10.36227/techrxiv.16577402.v3.
[42] B. Celiktas, "A hierarchical key assignment scheme for access control in cloud computing," Ph.D. dissertation, Dept. Appl. Inform., Istanbul Tech. Univ., Istanbul, Turkey, 2022.

IBRAHIM CELIKBILEK received the B.S. degree in electronics and computer from Fırat University, Turkey, in 2002, and the M.S. degree in cybersecurity engineering from Istanbul Sehir University, in 2016. He is currently pursuing the Ph.D. degree in cybersecurity engineering and cryptography program with the Institute of Informatics, Istanbul Technical University. His current research interests include cryptography, computational number theory, and secure software development.

BARIS CELIKTAS (Member, IEEE) received the B.S. degree in system engineering from National Defense University, in 2008, the M.S. degree in international relations from Karadeniz Technical University, in 2016, the M.S. degree in applied informatics from Istanbul Technical University, in 2018, and the Ph.D. degree in cybersecurity engineering and cryptography program from the Institute of Informatics, Istanbul Technical University, in 2022. Currently, he is an assistant professor position with the Computer Engineering Department. He is the Director of the Cyber Security Graduate Program with FMV Işık University. His research interests include cyber security, network security, cloud computing, and cryptography.

ENVER OZDEMIR (Senior Member, IEEE) received the B.S. degree in mathematics from Middle East Technical University, Turkey, in 2002, and the Ph.D. degree in mathematics from University of Maryland, College Park, MD, USA, in 2009. He was a Research Fellow with CCRG, NTU, Singapore, from 2010 to 2014. He is a Professor at Computer Engineering Department, Istanbul Technical University. He is also the Deputy (Executive) Director of National Center for High-Performance Computing. His research interests include cryptography, computational number theory, and network security.