

Efficient and Secure Distribution of Massive Geo-Spatial Data^{*†}

Hao Yuan and Mikhail J. Atallah
Department of Computer Science
Purdue University
West Lafayette, IN 47907, USA
{ yuan3, mja }@cs.purdue.edu

ABSTRACT

Modern geographic databases can contain a large volume of data that need to be distributed to subscribed customers. The data can be modeled as a cube, where typical dimensions include latitude, longitude, and time. One way of distributing the data consists of making freely available encrypted versions of selected subsets of the data, and giving each paying customer the decryption keys for their authorized subsets only. The total space for these encrypted versions should be close to linear in the size of the data, yet the subset for a customer can be an arbitrary orthogonal range of the data; there is a quadratic number of such subsets, and we do not know ahead of time which subset will be subscribed (so the almost linear, and a priori selected subsets, must be enough to exactly express any of the quadratic number of ranges that could possibly interest a customer). This is mainly a data structuring and algorithmic problem.

For a geo-spatial database in which the geography is modeled as an $m \times n$ grid of cells (with $m \geq n$), we provide a novel scheme that: (i) assigns a constant number of keys to a user; (ii) allows the user to derive the decryption key of her authorized rectangular area by using a constant number of inexpensive cryptographic operations (hash function computations); (iii) uses $O(mn \log^* m)$ public storage. This improves by a factor of $(\log \log m)^2$ the space complexity of the best previous result, while matching all its other performance characteristics. Our approach can also handle higher dimensional data efficiently, as long as each authorized region is an orthogonal range. The improved bounds are achieved using a combinatorial approach, and the only

^{*}Portions of this work were supported by Grant CNS-0627488 from the National Science Foundation, by Grant FA9550-09-1-0223 from the Air Force Office of Scientific Research, and by sponsors of the Center for Education and Research in Information Assurance and Security.

[†]An early version of this work was accepted in The 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS 2009).

cryptographic notion we use is that of a one-way hash function.

1. INTRODUCTION

1.1 Background

The storage capabilities of modern information systems are growing very fast. This makes it possible to store a huge amount of geo-spatial data with very high quality, e.g., high-resolution satellite images. Publishing such a large volume of data is a challenge, especially when coupled with access control enforcement and the support of data authentication by users, because of the potential expensive costs from server maintenance, personnel costs, etc. The third-party model of data publishing is designed to help distribute large amounts of data efficiently [25]. In this model, data can be distributed over public channels, like HTTP servers, P2P networks, wireless broadcasts, outsourced databases, etc. Providing access control in third-party data distribution is an important issue, as the data owner does not want users to obtain data outside of their authorized ranges.

In this paper, we study secure data publishing schemes for fine-grained temporal and geo-spatial data, like satellite images and traffic information. The importance of “fine-grained access control mechanisms permitting the precise release of location information to just the right parties under the right circumstances” was discussed in the summary of the NRC’s *IT roadmap to a geo-spatial future* [34]. It is one of the major future challenges in geo-spatial computing.

In the third-party distribution of geo-spatial data, we model the data to be an $m \times n$ grid of cells (with $m \geq n$). Temporal geo-spatial data gives rise to a 3-dimensional grid, whereas purely temporal data is 1-dimensional where each cell represents a timestamp. The geo-spatial data are distributed through untrusted third-party distributors in encrypted form. More specifically, each grid cell (data) is encrypted by the data owner, and can only be decrypted with a corresponding decryption key. A user can subscribe to a rectangular subgrid of cells. To access the subscribed data, the user may download the encrypted data from the third-party data distributor (e.g., outsourced databases), and then use the keys obtained from the data owner (or a separate trusted third-party key distributor) to decrypt the data that he or she is authorized to access.

Note that the model allows a user to download encrypted data that she is not authorized to access, but she should

not be able to decrypt the data until a subscription is made to that part of the data. This not only helps the user to download the data early when a fast channel is available (e.g., over a short-life P2P session), but also saves the data distributors’ access control costs (as it is now no need to enforce access control over encrypted data).

A naive approach would be to provide a subscribed user with all the decryption keys corresponding to the user’s authorized cell data. In this approach, the number of keys assigned to a user could be as large as $m \times n$, depending on the size of the user’s subscribed subgrid. Such a large number of decryption keys is undesirable (and in fact avoidable) when, for example, the user is using a weak hand-held device without enough space to store all the decryption keys — such a user prefers to have the ability to inexpensively compute any of the authorized decryption keys on demand. Therefore, key management is at the core of access control for third-party data distribution.

In the previous literature on key management, the main focus has been key management schemes for hierarchical access control systems [1, 4, 3, 37, 14, 18, 20, 30, 49]. Extensions of these schemes to support time-based policies were studied in [8, 17, 21, 28, 41, 42, 43, 46]. In a typical hierarchical access control system, users are assigned to access classes, and the access rights of a class are a superset of the access rights of every descendant of that class in the access hierarchy (see the hierarchically organized roles in RBAC models [35] for example). When considering temporal constraints, each user is given access rights for a specific time interval, in addition to having users organized in a hierarchy of classes. The access rights are realized by the specific keys for the corresponding access classes — a class has one key associated with it, that can efficiently derive any of the authorized decryption keys for that class. In the setting of geo-spatial data distributions, each subgrid of the data corresponds to an access class, and the number of subgrids (hence possible access classes) is as large as $O(m^2 n^2)$.

The literature of key management for access control suggests key derivation mechanisms to handle such a large number of access classes and keys. In schemes that are based on key derivations, each user is given a set of secret keys that it can use to derive only the keys of access classes that it is authorized to access. The schemes assume a (preferably small) amount of information that is public, i.e., accessible to all users through the Internet. In the access control of geo-spatial data, each grid cell corresponds to a base class, and the access key for that base class is the decryption key for accessing the cell’s data.

The evaluation criteria for a geo-spatial key derivation scheme are:

- the number of secret keys that are assigned to a user;
- the amount of public storage in the server that is used to help key derivations;
- the amount of computation it takes a user to derive a decryption key of a data cell in her subscribed subgrid.

In this paper, we develop efficient key assignment and derivation schemes for access control of temporal and geo-spatial

Source	Private Storage	Key Derivation	Public Storage
[6]	$O(1)$	$O(1)$	$O(m \log \log m \log^* m)$
[6]	$O(1)$	$O(\log^* m)$	$O(m \log \log m)$
this	$O(1)$	$O(1)$	$O(m \log^* m)$
this	$O(\log^* m)$	$O(1)$	$O(m)$

Table 1: Performance comparisons for temporal data access control.

Source	Private Storage	Key Derivation	Public Storage
[5]	$O(1)$	$O(1)$	$O(mn(\log \log m)^2 \log^* m)$
this	$O(1)$	$O(1)$	$O(mn \log^* m)$
this	$O(\log^* m)$	$O(1)$	$O(mn)$

Table 2: Performance comparisons for geo-spatial data access control.

Source	Private Storage	Key Derivation	Public Storage
this	$O(1)$	$O(1)$	$O(N \log^* N)$
this	$O(\log^* N)$	$O(1)$	$O(N)$

Table 3: Summarized results of our high dimensional schemes when d is fixed. N is the total number of data cells.

data that are low in all of the above cost metrics, and that improve on the previous state of the art.

1.2 Our Contributions

Table 1 summarizes the results of our key management schemes for temporal data. The temporal data are a set of data items, each of which is labeled by a timestamp. Users can subscribe to data items that belong to a time interval. To be consistent with the grid-based notation for geo-spatial access control, the temporal data are represented by an $m \times 1$ grid, and each subgrid represents a time interval. Our proposed schemes are compared to the previous best-known work for temporal access control by Atallah et al.[6] in Table 1. Here, we do not assume the existence of a user hierarchy, but our scheme could be incorporated into existing hierarchy-based key management schemes with the general technique of De Santis et al. [36] or Atallah et al. [6].

The centerpiece of this paper is the key derivation scheme we provide for geo-spatial grids, because it uses a novel data structure to achieve the following characteristics for a grid composed of $m \times n$ data cells (where $m \geq n$):

- To derive the decryption keys of cell data in an arbitrary rectangular subgrid, a user is required to store only a constant number of keys.
- Key derivation within the subscribed rectangle subgrid involves a constant number of operations, where all the cryptographic operations are “lightweight” (sim-

ple cryptographic hashes, and no modular exponentiation).

- The public storage to assist key derivations is almost linear: $O(mn \log^* m)$.

Previous best-known key management scheme that achieve constant number of secret keys and constant number of key derivation operations are due to Atallah et al. [5]. Their scheme required $O(mn(\log \log m)^2 \log^* m)$ public storage space. Therefore, we have improved the space complexity by a factor of $(\log \log m)^2$. Additionally, our scheme is balanced in the sense that all resource consumption such as the client’s private storage, computation to derive keys, and the server public storage are minimized, with tradeoffs being possible. This allows the scheme to work even with very weak clients and not to burden the server with excessive storage. A summary of our results is shown in Table 2. Our schemes are constructed by a new approach that is different from the previous work.

Our schemes can be extended to support access control for higher dimensional data (e.g., OLAP data cube [19]). Access control for high dimensional data is important. For example, geo-spatial data may be continuously updating (e.g., real-time traffic information), and the data owner/collector wants to impose a time constraint on users’ subscriptions. In this case, a user is only allowed to decrypt data items in a subgrid for a specific time interval. Assume that time is divided into different time slots, and each data item is labeled (in addition to its spatial coordinates) by one of the time slots, then a time interval can be represented by a set of contiguous time slots. This is indeed an application of 3d grids (combined temporal and geo-spatial).

The security requirements of the key management schemes are: Given a user’s secret keys, the user \mathcal{U} : (i) can generate all keys for her designated subgrid $V_{\mathcal{U}}$ (completeness) and (ii) cannot generate a key outside of her $V_{\mathcal{U}}$ (soundness). Our schemes use the key derivation method of [4] as a basic building block, inheriting the security properties of the schemes in [4], i.e., our schemes are secure against key recovery (or key indistinguishability with slight modification) and collusion of multiple users. The security against collusion attack means that if any subset of users collude, then they are only able to derive the decryption keys of the union of their subscribed data items.

Roadmap. This paper follows the style and notation of [6, 5], on which it builds and improves, and is organized as follows. Section 2 provides a brief review of related work. Section 3 gives a formal problem definition, and describes building blocks that are used later in this paper. Section 4 presents the improved schemes for temporal data access control. New schemes for geo-spatial data access control are presented in Section 5. Section 6 discusses the security of the schemes. Section 7 concludes.

2. RELATED WORK

The third-party models for distribution of data have been well studied, many of which are in the form of outsourced databases [25]. Most of the research in outsourced databases focus on integrity verifications of the query results and some privacy issues [22, 31, 27, 32, 33, 29, 24, 15, 16, 44, 38, 7,

23, 13].

The literature on time-invariant key assignment (KA) schemes in a user hierarchy is extensive, and we do not survey it in this paper – for an overview of such publications, see, e.g., [3] and [20]. While the list of publications on time-invariant KA schemes is very large, the number of publications that consider time-based policies and provide schemes for them is rather modest. The time-based setting and the first scheme was introduced by Tzeng [41], and many subsequent schemes were developed [48, 28, 17, 46, 40, 47, 21, 8, 43]. The work of Ateniese et al. [8] was the first result that provided a formal framework for time-based hierarchical KA schemes and gave solutions that are provably secure with respect to key indistinguishability. The work of De Santis et al. [36] lists solutions with different performance parameters, and their schemes are provably secure with respect to key indistinguishability. Concurrently and independently, Atallah et al. [6] presented solutions with better performance than the work of [36]. Both of these works [36, 6] have described a general technique to incorporate any temporal data key management scheme into hierarchical key management schemes.

Recently, geo-spatial access control models were proposed in [9, 11, 2]. These models, however, concentrate on policy specification and not on key assignment or derivation mechanisms. Atallah et al. [5] first provided an efficient key management scheme that implements geo-spatial access control policies, and their scheme can be incorporated into existing hierarchy-based key management schemes using techniques similar to [36, 6]. The recent work of Srivatsa et al. [39] considers the specific application of location-based broadcast services.

3. PRELIMINARIES

In Section 3.1, we follow the presentation from [5, 6] to list the requirements of the key management scheme for the temporal and geo-spatial access controls. Section 3.2 and 3.3 review a hash-based key derivation technique that will be used as a building block of our key management schemes.

3.1 Problem Definitions

In a temporal access control model, the data items are labeled with m timestamps, denoted by $T = \{t_1, t_2, \dots, t_m\}$. For each timestamp t_i , we assign a decryption key k_{t_i} . Denote the set of decryption keys to be $K = \{k_{t_1}, k_{t_2}, \dots, k_{t_m}\}$. Assume that a user \mathcal{U} is authorized to access the system for a set of contiguous timestamps $T_{\mathcal{U}} = \{t_{\text{start}}, t_{\text{start}+1}, \dots, t_{\text{end}}\}$, where indices “start” and “end” are the parameters to identify the user’s authorized time interval. With such access rights, \mathcal{U} should be able to compute the decryption keys $K_{T_{\mathcal{U}}} \subseteq K$, where $K_{T_{\mathcal{U}}} = \{k_{t_i} \mid t_i \in T_{\mathcal{U}}\}$.

Similarly, in a two dimensional geo-spatial access control model, we have an m by n grid (where $m \geq n$), and each grid node $v_{i,j}$ ($1 \leq i \leq m$ and $1 \leq j \leq n$) is assigned with a decryption key $k_{v_{i,j}}$ to allow a user to decrypt the content that is associated with the node. Denote the set of the grid nodes to be V_0 . Typically, a user \mathcal{U} is authorized to access a rectangular region $V_{\mathcal{U}}$ of the grid, where

$$V_{\mathcal{U}} = \{v_{i,j} \mid x_1(\mathcal{U}) \leq i \leq x_2(\mathcal{U}) \text{ and } y_1(\mathcal{U}) \leq j \leq y_2(\mathcal{U})\}.$$

The parameters $x_1(\mathcal{U}), x_2(\mathcal{U}), y_1(\mathcal{U})$ and $y_2(\mathcal{U})$ are used to identify the subgrid $V_{\mathcal{U}}$. Denote $K_{V_{\mathcal{U}}} = \{k_{v_i,j} \mid v_i,j \in V_{\mathcal{U}}\}$.

Since the temporal case is just the one dimensional version of the spatial case, to make our notation consistent, we use a graph node v_i to represent t_i , and define $V_{\mathcal{U}} = \{v_i \mid x_1(\mathcal{U}) \leq i \leq x_2(\mathcal{U})\}$ where $x_1(\mathcal{U})$ and $x_2(\mathcal{U})$ are the indices of the “start” and “end” time. The notation can be generalized to support access control for high dimensional data.

A key assignment scheme assigns to each user \mathcal{U} some private information, denoted by $S_{V_{\mathcal{U}}}$, which can be used to compute $K_{V_{\mathcal{U}}}$ with the help of the public information. The private information is assigned by a central authority CA. The user interacts with the CA only once, to obtain its private information $S_{V_{\mathcal{U}}}$. A naive key assignment scheme can simply assign all the decryption keys of $K_{V_{\mathcal{U}}}$ to the user \mathcal{U} , but in this case $|S_{V_{\mathcal{U}}}|$ can be as large as mn . Much more efficient key assignment schemes can be developed to optimize the size of $S_{V_{\mathcal{U}}}$, which will be discussed in the later sections.

The security requirement of a key management scheme is standard — we require the properties of completeness and soundness to hold, e.g., in the setting of the geo-spatial access control:

Completeness A user with subscription to a rectangular area $V_{\mathcal{U}}$ is able to compute the decryption key for each cell within $V_{\mathcal{U}}$.

Soundness Any coalition of users with access to rectangle areas $V_{\mathcal{U}_1}, \dots, V_{\mathcal{U}_k}$ is unable to obtain access to any cell other than those in $V_{\mathcal{U}_1} \cup \dots \cup V_{\mathcal{U}_k}$.

In this paper, we propose key assignment (KA) schemes that are secure against *key recovery*. With slight modification (as in [4, 37]), the scheme can be adapted to be secure against key indistinguishability. In both cases, the schemes are secure against collusion attack.

3.2 Key Derivation for Graphs

In this work, we use the key derivation technique of [4] to design the key assignment scheme. The key derivation technique of [4] works for any directed acyclic graph (DAG) $G = (V, E)$, where V is the node set of the graph, and E is the edge set of the graph. It consists of two algorithms: **Set**, an algorithm to setup the system; **Derive**, an algorithm to derive a key. The **Set** algorithm assigns secret keys to the nodes of the graph and computes public information associated with its edges. The **Derive** algorithm, given a node $v \in V$, its secret key k_v and another node $w \in V$, computes the secret key of w using the public information about G as long as w is a descendant of v in the graph.

In what follows, $F : \{0, 1\}^{\kappa} \times \{0, 1\}^* \rightarrow \{0, 1\}^{\kappa}$ denotes a family of pseudo-random functions (PRFs) that, on input a κ -bit key and a string, outputs a κ -bit string that is indistinguishable from random. Note that a PRF can be efficiently implemented using HMAC [10] or CBC MAC constructions.

The **Set** and **Derive** algorithms are then as follows:

- **Set**($1^{\kappa}, G$): For each node $v \in V$, select a random secret key $k_v \in \{0, 1\}^{\kappa}$. For each node $v \in V$, select a unique label $l_v \in \{0, 1\}^*$ and make it publicly available. For each edge $(v, w) \in E$, compute $y_{v,w} = k_w \oplus F(k_v, l_w)$, where \oplus denotes bitwise XOR, and make it publicly available.
- **Derive**(v, w, k_v): Let $(v, w) \in E$, i.e., v is a parent node of w . Then given k_v and the above-mentioned public information, derivation of the key k_w can be performed as $k_w = F(k_v, l_w) \oplus y_{v,w}$, where l_w and $y_{v,w}$ are publicly available. More generally, if there is a directed path between nodes v and u in G , then u 's key can be derived from v 's key along the path.

In other words, there is a public label associated with every node in the graph, and there is a public information associated with every edge in the graph. This public information is what allows users to derive appropriate keys.

3.3 Shortcut techniques

Our constructions use the so-called shortcut edges: a *shortcut edge* is an edge that is not in the original graph G but is in the transitive closure of G . Such edges are added to G for performance reasons. Note that addition of shortcut edges does not affect partial order relationship between the nodes.

Shortcutting schemes for a directed chain graph will be used as a building block. A directed chain graph is a DAG with directed edges $\{(v_i, v_{i+1}) \mid 1 \leq i < |V|\}$. There exists a scheme [4] that adds at most $O(m \log^* m)$ edges to a directed chain, so that the shortest path between any two nodes is at most 4. We use **4HS** (short for 4-Hop Scheme) to denote such a scheme. Throughout this work, $\text{DerTime4HS}(m) = 4$ and $\text{PubSp4HS}(m) = O(m \log^* m)$ are used to represent the **4HS**'s complexities of the key derivation time and the public storage respectively. Note that the number of secret keys is 1 in this case.

4. ACCESS CONTROL FOR TEMPORAL DATA

Our key assignment scheme for temporal data uses Atallah et al.'s scheme [5] as a subroutine. In their scheme, the size of the public storage is $O(m \log \log m \log^* m)$, and a user \mathcal{U} is assigned a constant number of secret keys which can be used to derive the decryption key of any timestamp in \mathcal{U} 's authorized time interval in constant time. In Section 4.1, using a technique of Yao [45], we improve the time-based key assignment scheme to use only $O(m(\log^* m)^2)$ public storage, while the number of the secret keys that a user holds and the key derivation time remain constant. Section 4.2 further improves the public space complexity to be $O(m \log^* m)$ based on the result of Section 4.1. The result can be improved further: a scheme that uses $O(m \log^* \log^* m)$ public space is obtained in Section 4.4, and this scheme will be used as a base scheme for our geo-spatial schemes. Section 4.3 gives a scheme that uses linear public storage (i.e., $O(m)$), while the number of secret keys that are assigned to a user is $O(\log^* m)$.

4.1 $O(m(\log^* m)^2)$ -Space Scheme

In this subsection, we will give temporal scheme 1, which is used as a tool for the temporal scheme 2 that will be discussed at Section 4.2.

4.1.1 Building Data Structures

Since we will use Atallah et al.'s data structures from [6] as a basic scheme, we will first give an interface for calling their data structures building algorithm:

BuildDS_Temporal_Basic(m, V_0): Given a set of timestamps $V_0 = \{v_1, v_2, \dots, v_m\}$, build a data structure of size $O(m \log \log m \log^* m)$ that supports: $O(1)$ number of secret keys can be assigned to a user \mathcal{U} so that she can derive the decryption keys $K_{V_{\mathcal{U}}}$ in $O(1)$ time, where her authorized timestamps $V_{\mathcal{U}}$ is in the contiguous form (i.e., a time interval). We use **NumKeysT0**(m), **DevTimeT0**(m) and **PubSpT0**(m) to represent the basic scheme's complexities of the number of secret keys, the key derivation time and the public storage space respectively.

The following algorithm builds data structures for temporal scheme 1.

Algorithm BuildDS_Temporal_1(m, V_0)

Input: m nodes $V_0 = \{v_1, v_2, \dots, v_m\}$ that represent contiguous timestamps;

Return: a tree node v ;

1. Create a tree node v , at which we will store the data structures to solve the key management problem for nodes set V_0 .
2. If $m = 1$, then associate v_1 (the only graph node in V_0) to the tree node v by setting $\text{TN}(v_1) \leftarrow v$; Return the tree node v .
3. Divide the timestamps into $m/\log m$ blocks¹, where each block $B_i(v)$ ($1 \leq i \leq m/\log m$) is defined by $B_i(v) = \{v_j \mid (i-1)\log m < j \leq i\log m\}$.
4. For each $B_i(v)$, create a super-node \hat{v}_i (also denoted by $\text{SN}_i(v)$) and link an arc from \hat{v}_i to each node in $B_i(v)$. Assign a random key $k_{\hat{v}_i}$ to \hat{v}_i . This allow a user to derive any key of $K_{B_i(v)}$ in constant time as long as the user has the key $k_{\hat{v}_i}$. See Figure 1 for example.
5. Call **BuildDS_Temporal_Basic** on top of $\text{Coarse}(v) = \{\hat{v}_i \mid 1 \leq i \leq m/\log m\}$.
6. For each $B_i(v)$, create a set of R-nodes

$$B_i^R(v) = \left\{ v_j^R \mid (i-1)\log m < j \leq i\log m \right\},$$

and assign random keys to the nodes in $B_i^R(v)$. For each v_j^R , link an arc from v_j^R to v_j . For each v_j^R where $(i-1)\log m < j < i\log m$, link an arc from v_j^R to v_{j+1}^R . Create shortcut edges based on the 4-Hop Scheme (see Section 3.3) on top of the subgraph induced by the node set $B_i^R(v)$. This will allow any user who is assigned the key $k_{v_p^R}$ (where $(i-1)\log m < p \leq i\log m$) to derive $k_{v_j^R}$ for any j such that $p \leq j \leq i\log m$ within

¹To avoid unnecessarily cluttering the exposition, we avoid using the floor and ceiling functions in this paper. It is easily seen that there is no loss of generality in doing so.

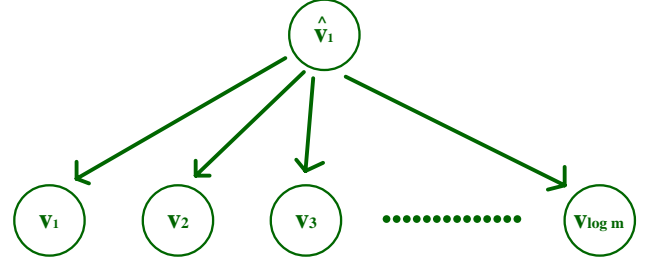


Figure 1: Example on $B_1(v)$ for step 4 of Algorithm BuildDS_Temporal_1.

constant number of operations, and then subsequently derive the decryption key k_{v_j} by one more operation. See Figure 2 for an example.

7. For each node set $B_i(v)$, create a set of L-nodes

$$B_i^L(v) = \left\{ v_j^L \mid (i-1)\log m < j \leq i\log m \right\},$$

and assign random keys to the nodes in $B_i^L(v)$. For each v_j^L , link an arc from v_j^L to v_j . For each v_j^L where $(i-1)\log m + 2 \leq j \leq i\log m$, link an arc from v_j^L to v_{j-1}^L . Create shortcut edges based on the 4-Hop Scheme (see Section 3.3) on top of the subgraph induced by the node set $B_i^L(v)$. This will allow any user who is assigned the key $k_{v_p^L}$ (where $(i-1)\log m < p \leq i\log m$) to derive $k_{v_j^L}$ for any j such that $(i-1)\log m < j \leq p$ within constant number of operations, and then subsequently derive the decryption key k_{v_j} by one more operation. See Figure 2 for an example.

8. Call **BuildDS_Temporal_1**($|B_i(v)|, B_i(v)$) for $1 \leq i \leq m/\log m$. Assume that the returned tree node is v_c after each call, then make v_c a child of v .
9. Return the tree node v .

The public storage complexity of Step 5 is proportional to

$$\begin{aligned} \text{PubSpT0}(|\text{Coarse}(v)|) &= \text{PubSpT0}(m/\log m) \\ &= \left(\frac{m}{\log m}\right) \cdot \log \log \left(\frac{m}{\log m}\right) \cdot \log^* \left(\frac{m}{\log m}\right) = O(m), \end{aligned}$$

and the space complexity of step 6 and 7 is proportional to

$$\begin{aligned} \sum_{1 \leq i \leq m/\log m} \text{PubSp4HS}(|B_i(v)|) &= \frac{m}{\log m} \cdot \text{PubSp4HS}(\log m) \\ &= \frac{m}{\log m} \cdot (\log m) \log^*(\log m) = O(m \log^* m). \end{aligned}$$

All other steps are $O(m)$. Therefore, the total space of the improved data structure, **PubSpT1**(m), satisfies the following recurrence,

$$\begin{aligned} \text{PubSpT1}(m) &\leq \frac{m}{\log m} \text{PubSpT1}(\log m) + c_1 \cdot m \log^* m; \\ \text{PubSpT1}(1) &= c_2; \end{aligned}$$

where c_1 and c_2 are constants. Thus, we have $\text{PubSpT1}(m) = O(m(\log^* m)^2)$. The recursion tree formed by the tree nodes is of size $O(m)$ as there are m leaves.

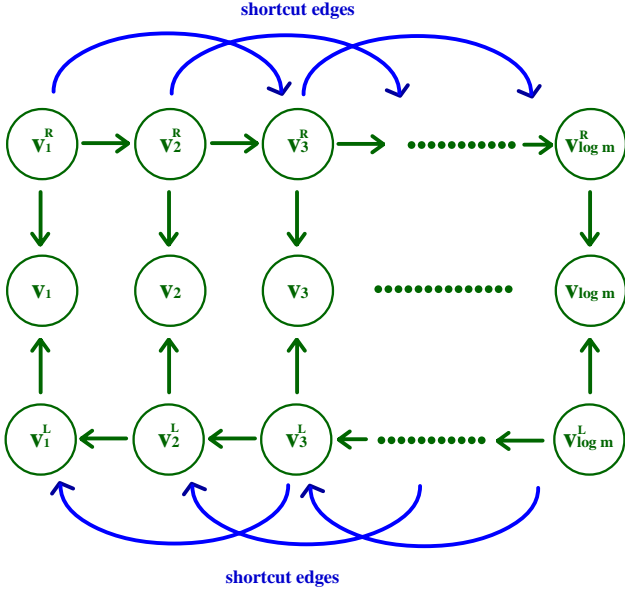


Figure 2: Example on $B_1(v)$ for step 6 and 7 of Algorithm `BuildDS_Temporal_1`.

4.1.2 Key Assignment and Derivation

With the data structure built in Section 4.1.1, we can use the following key assignment scheme to assign a constant number of secret keys to a user \mathcal{U} who is subscribed to access the data items in the time interval $V_{\mathcal{U}} = \{v_i \mid x_1(\mathcal{U}) \leq i \leq x_2(\mathcal{U})\}$ where $x_1(\mathcal{U})$ and $x_2(\mathcal{U})$ are the indices of the “start” and “end” time. We call `AssignKeys_Temporal_1` by setting parameter v to be the root node of the data structures built from Section 4.1.1.

Algorithm `AssignKeys_Temporal_1(v, V_{\mathcal{U}})`

1. If $|V_{\mathcal{U}}| = 1$, assign to \mathcal{U} the key of the only timestamp in $V_{\mathcal{U}}$, then return.
2. If $V_{\mathcal{U}} \subseteq B_i(v)$ for some i , then find the child node v_c of v that corresponds to $B_i(v)$, and call `AssignKeys_Temporal_1(v_c, V_{\mathcal{U}})`. Return after the call.
3. If $V_{\mathcal{U}} \not\subseteq B_i(v)$ for any i , then $V_{\mathcal{U}}$ overlaps with more than one block. Let i_1 and i_2 be the indices of the first and last blocks that $V_{\mathcal{U}}$ overlaps with, i.e., $V_{\mathcal{U}} \cap B_i(v) \neq \emptyset$ for $i_1 \leq i \leq i_2$, and $V_{\mathcal{U}} \cap B_{i_1-1}(v) = \emptyset$, $V_{\mathcal{U}} \cap B_{i_2+1}(v) = \emptyset$.
4. To allow \mathcal{U} to derive the keys in $B_i(v)$ for $i_1 < i < i_2$, we call the key assignment algorithm of the basic scheme (used to build data structures for `Coarse(v)`) to assign a constant number of secret keys to \mathcal{U} , which will allow her to derive any key of $\{k_{v_i} \mid i_1 < i < i_2\}$ in constant time, and subsequently derive k_{v_j} for $i_1 \log m < j \leq (i_2 - 1) \log m$ by one more operation.
5. To allow \mathcal{U} to derive the keys in $B_{i_1}(v) \cap V_{\mathcal{U}}$, we assign the key $v_{x_1}^R$ to \mathcal{U} where x_1 is the index of the leftmost node in $V_{\mathcal{U}}$. The 4-Hop Scheme data structures built

for $B_{i_1}^R(v)$ allow \mathcal{U} to derive the key for v_j^R within constant time, and subsequently derive the key for v_j by one more operation, where $x_1 \leq j \leq i_1 \log m$.

6. To allow \mathcal{U} to derive the keys in $B_{i_2}(v) \cap V_{\mathcal{U}}$, we assign the key $v_{x_2}^L$ to \mathcal{U} where x_2 is the index of the rightmost node in $V_{\mathcal{U}}$. The 4-Hop Scheme data structures built for $B_{i_2}^L(v)$ allow \mathcal{U} to derive the key for v_j^L within constant time, and subsequently derive the key for v_j by one more operation, where $(i_2 - 1) \log m < j \leq x_2$.

In step 4,5 and 6 of `AssignKeys_Temporal_1`, a constant number of secret keys are assigned to \mathcal{U} , and the time to derive any decryption key of $K_{V_{\mathcal{U}}}$ is also constant.

A naive implementation of the key assignment algorithm will take $O(\log^* m)$ recursions for step 2 to reach the recursion level (i.e., the tree node v) where $V_{\mathcal{U}}$ overlaps with at least two blocks. To speed up the process, we can directly find the desired tree node v by computing the nearest common ancestor of $\text{TN}(v_{x_1(\mathcal{U})})$ and $\text{TN}(v_{x_2(\mathcal{U})})$ in the recursion tree within constant time, and all we need is a linear time/space preprocessing (see [26]).

4.2 $O(m \log^* m)$ -Space Scheme

To further improve the public storage complexity, we reapply the technique of Section 4.1 to get our temporal scheme 2. The new scheme is almost the same as temporal scheme 1, with only the following differences:

- The block size is now chosen to be $\log^* m$.
- In step 5, apply `BuildDS_Temporal_1` on `Coarse(V_0)` instead of `BuildDS_Temporal_Basic`.
- Do not recursively build the data structures (i.e., `BuildDS_Temporal_2`) for each block (Step 8). Instead, apply `BuildDS_Temporal_1` on each block.

The public storage complexity for the `Coarse(V_0)` is now proportional to

$$\text{PubSpT1}(|\text{Coarse}(V_0)|) = \text{PubSpT1}(m/\log^* m) = O(m \log^* m),$$

and the total space of the data structures for each block is proportional to

$$\begin{aligned} & \sum_{1 \leq i \leq m/\log^* m} \text{PubSpT1}(|B_i|) \\ & = (m/\log^* m) \cdot \text{PubSpT1}(\log^* m) = O(m \log^* m). \end{aligned}$$

Therefore, the space complexity of the improved data structures is $\text{PubSpT2}(m) = O(m \log^* m)$. The key assignment and derivation schemes are created accordingly, and the number of secret keys and the key derivation time remain constant.

4.3 Linear Public Storage Scheme

In this subsection, we will give our temporal scheme 3, which requires $O(m)$ public storage, and allow a user \mathcal{U} who is assigned $O(\log^* m)$ secret keys to be able to derive any of her authorized decryption keys in constant time.

4.3.1 Linear Data Structures

We build the data structures as follows.

Algorithm BuildDS_Temporal_3(V_0)**Input:** $V_0 = \{v_1, v_2, \dots, v_m\}$

1. Divide V_0 into $m/\log^* m$ blocks, each of which is of size $\log^* m$. Denote each block by B_i ($1 \leq i \leq m/\log^* m$) where $B_i = \{v_j \mid (i-1)\log^* m < j \leq i\log^* m\}$.
2. For each B_i , create a super-node \hat{v}_i and assign a random key $k_{\hat{v}_i}$ to \hat{v}_i . Link an arc from \hat{v}_i to each node in B_i .
3. Apply BuildDS_Temporal_2 on $\text{Coarse}(V_0) = \{\hat{v}_i \mid 1 \leq i \leq m/\log^* m\}$ to support temporal access control for a $V_{\mathcal{U}}$ that is the union of one or more contiguous blocks.

Step 1 and 2 require $O(m)$ time and space, and the storage complexity of Step 3 is proportional to

$$\text{PubSpT2}(|\text{Coarse}(V_0)|) = \text{PubSpT2}(m/\log^* m) = O(m).$$

Therefore, the total space complexity of this scheme is $O(m)$.

4.3.2 Key Assignment and Derivation

The key assignment and derivation schemes are created accordingly as follows:

Algorithm AssignKeys_Temporal_3($V_{\mathcal{U}}$)

1. If $V_{\mathcal{U}} \subseteq B_i$ for some i , then just assign the keys $K_{V_{\mathcal{U}}}$ to \mathcal{U} and return.
2. If $V_{\mathcal{U}} \not\subseteq B_i$ for any i , then $V_{\mathcal{U}}$ overlaps with more than one block. Let i_1 and i_2 be the indices of the first and last blocks that $V_{\mathcal{U}}$ overlaps with, i.e., $V_{\mathcal{U}} \cap B_i \neq \emptyset$ for $i_1 \leq i \leq i_2$, and $V_{\mathcal{U}} \cap B_{i_1-1} = \emptyset$, $V_{\mathcal{U}} \cap B_{i_2+1} = \emptyset$.
3. To allow \mathcal{U} to derive the keys in B_i for $i_1 < i < i_2$, we call the key assignment algorithm of the **temporal scheme 2** (used to build data structures for $\text{Coarse}(V_0)$) to assign a constant number of secret keys to \mathcal{U} , which allow her to derive any key of $\{k_{\hat{v}_i} \mid i_1 < i < i_2\}$ in constant time, and subsequently derive k_{v_j} for $i_1 \log^* m < j \leq (i_2 - 1)\log^* m$ by one more operation.
4. Also assign to \mathcal{U} the decryption keys $\{k_v \mid v \in (B_{i_1} \cup B_{i_2}) \cap V_{\mathcal{U}}\}$.

Step 1 and 4 assign at most $O(\log^* m)$ number of decryption keys and there is no need to do any key derivation. Step 3 assigns a constant number of secret keys to \mathcal{U} , and the key derivation time remains constant as shown in the algorithm.

4.4 $O(m \log^* \log^* m)$ -Space Scheme

To get the new **temporal scheme 4**, we reapply the technique of Section 4.2. The new scheme is modified from **temporal scheme 1** (in Section 4.1) as following:

- The block size is now chosen to be $\log^* m$.
- In step 5, apply BuildDS_Temporal_2 on $\text{Coarse}(V_0)$ instead.
- Do not recursively build the data structures for each block (Step 8). Instead, apply BuildDS_Temporal_2 on each block.

Based on these modifications, the space complexity for $\text{Coarse}(V_0)$ becomes

$$\begin{aligned} & \text{PubSpT2}(|\text{Coarse}(V_0)|) \\ &= \text{PubSpT2}(m/\log^* m) = \frac{m}{\log^* m} \cdot \log^* \frac{m}{\log^* m} = O(m), \end{aligned}$$

and the total space of the data structures for each block is proportional to

$$\begin{aligned} & \sum_{1 \leq i \leq m/\log^* m} \text{PubSpT2}(|B_i|) \\ &= m/\log^* m \cdot \text{PubSpT2}(\log^* m) = O(m \log^* \log^* m). \end{aligned}$$

Therefore, the public storage complexity of the **temporal scheme 4** is

$$\text{PubSpT4}(m) = O(m \log^* \log^* m).$$

The key assignment and derivation schemes are created accordingly as in the **temporal scheme 2**. The number of secret keys and the key derivation time are still constants.

5. ACCESS CONTROL FOR GEO-SPATIAL DATA

In this section, we will present a new scheme for geo-spatial access control using any exiting temporal access control scheme as a base scheme. The technique we used is similar to the dimension reduction technique of Chazelle and Rosenberg [12].

5.1 Building Data Structures

Any temporal scheme can be considered as a scheme for an $m \times 1$ grid. We will use **temporal scheme 4** given in Section 4.4 as the **base scheme** for our constructions in this section. The following algorithm is used to build the data structures.

Algorithm BuildDS_GeoSpatial_New(V_0)**Input:** $m \times n$ grid nodes $V_0 = \{v_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n\}$

1. Let D_j denote the node set $D_j = \{v_{i,j} \mid 1 \leq i \leq m\}$, where $1 \leq j \leq n$.
2. For each $1 \leq j \leq n$, build the **base scheme** for each D_j , denote the resulting DAG by G_j (i.e., the data structures). Let V_j denote the node set of G_j , and $V_j(x)$ denote the x_{th} node of it. Note that the ordering of the nodes in V_j must satisfy the following **key property**:

For any i_1, i_2, j_1, j_2 , let

- \mathcal{U}_1 be a user whose authorized region is $\{v_{i,j_1} \mid i_1 \leq i \leq i_2\}$;
- \mathcal{U}_2 be a user whose authorized region is $\{v_{i,j_2} \mid i_1 \leq i \leq i_2\}$;
- X_1 be the indices of the nodes of G_{j_1} whose corresponding keys are assigned to \mathcal{U}_1 by the **base scheme**, i.e., the keys are $\{k_{V_{j_1}(x)} \mid x \in X_1\}$;
- X_2 be the indices of the nodes of G_{j_2} whose corresponding keys are assigned to \mathcal{U}_2 by the **base scheme**, i.e., the keys are $\{k_{V_{j_2}(x)} \mid x \in X_2\}$;

then we must have $X_1 = X_2$.

The above property can be easily satisfied by numbering the graph nodes according to their order of creation.

3. For each $1 \leq x \leq |V_1|$, let $F_x = \{V_j(x) \mid 1 \leq j \leq n\}$; build data structures of the **base scheme** on top of F_x . Denote the data structures by G_{F_x} .

Public Storage Complexity Analysis

The public storage complexity of step 2 is proportional to

$$\sum_{1 \leq j \leq n} |V_j| = n \cdot \text{PubSpT4}(m) = O(mn \log^* \log^* m),$$

and the complexity of step 3 is

$$\begin{aligned} \sum_x |G_{F_x}| &= \sum_x \text{PubSpT4}(|F_x|) \\ &= |V_1| \cdot \text{PubSpT4}(n) = \text{PubSpT4}(m) \cdot \text{PubSpT4}(n) \\ &= O(m(\log^* \log^* m)) \cdot O(n(\log^* \log^* n)) \\ &\leq O(mn(\log^* \log^* m)^2) \leq O(mn \log^* m). \end{aligned}$$

Therefore, the total space complexity of the algorithm is $O(mn \log^* m)$.

5.2 Key Assignment and Derivation

The following key assignment algorithm is based on the **key property** of step 2 of algorithm `BuildDS_GeoSpatial_New`. In the key assignment algorithm, we use constant c to denote the maximum possible number of secret keys that are assigned to a user by the **base scheme**.

Algorithm `AssignKeys_GeoSpatial_New`(V_U)

Input: $V_U = \{v_{i,j} \mid x_1 \leq i \leq x_2, y_1 \leq j \leq y_2\}$, identified by x_1, x_2, y_1, y_2 .

1. Create a pseudo-user U' with authorized subgrid $V_{U'} = \{v_{i,1} \mid x_1 \leq i \leq x_2\}$.
2. Let X be the indices of the nodes from G_1 that are associated with the secret keys assigned to U' by the **basic scheme**, i.e., the keys assigned to U' are $\{k_{V_1(x)} \mid x \in X\}$. Note that we have $|X| = c$ according to the definition.
3. For each $x \in X$, use the key assignment algorithm of the **base scheme** to assign keys to U from the data structures G_{F_x} , where those keys allow U to derive keys for $\{V_j(x) \mid y_1 \leq j \leq y_2\}$.

The total number of keys that are assigned to U (in step 3) is c^2 (still a constant).

The key derivation can be done as follows: Assume that U wants to derive a key $v_{x_t, y_t} \in V_U$, where x_t and y_t are called the “target” indices, then

- First, find the index x from X such that: if U has the key for $V_1(x)$, and she can derive the key for $v_{x_t, 1}$ in constant time.
- Second, use the secret keys of U to derive the key for $V_{y_t}(x)$ in constant time.
- Third, use the key for $V_{y_t}(x)$ to derive the key for v_{x_t, y_t} in constant time.

The correctness of the above key derivation algorithm is based on the **key property** discussed in the last subsection. It can be seen that the total time complexity for the key derivation is still constant.

5.3 Linear Space Scheme

A linear space scheme that achieves the following performance characteristics is possible:

- assigns only $O(\log^* m)$ secret keys to a user;
- allows the user to derive appropriate keys in constant time.

We sketch the basic idea as follows : Use the temporal scheme from Section 4.4 to refine the scheme of Section 4.3, which will result in a temporal scheme that: (i) use $O(m)$ public storage; (ii) $O(\log^* \log^* m)$ secret keys are assigned to a user; (iii) the key derivation time is constant. Use this scheme as the **base scheme** for the data structure building algorithm and the key assignment/derivation algorithm in this section. Then the space complexity will be $O(mn)$ according to the complexity analysis of Section 5.1; and the number of secret keys that are assigned to a user is $O((\log^* \log^* m)^2)$ according to the analysis in Section 5.2, which is bounded by $O(\log^* m)$.

5.4 High Dimension Extensions

To achieve the claimed complexity summarized in Table 3 for a d -dimensional problem whose number of cells (grid nodes) is N , we use the dimension reduction technique (as in Section 5.1) to recursively reduce the dimensionality down to the temporal (1D) case. The resulting data structure will have $O(N(\log^* \log^* N)^d)$ space, which is $O(N \log^* N)$ when d is fixed.

Note that any temporal (1D) scheme can be used as a base scheme in our construction, even when the base scheme is not graph-based (i.e., iterative key derivation) and completely different from ours.

6. SECURITY

In this section, we sketch the proof of security (completeness and soundness defined in Section 3.1). Note that in the proof below, each “assigned node” corresponds to an “assigned key” in the key assignment algorithms, because of the one-to-one correspondence between the keys and the graph nodes.

THEOREM 1. *The proposed schemes satisfy the completeness requirement.*

PROOF. This is because of the nature of the graph-based key derivation technique: A user is assigned the secret keys to access a small number of graph nodes, and then he or she can derive the decryption keys for all the nodes (or cells) that are reachable from any of the assigned nodes. From our proposed graph constructions, it is straightforward to show that the reachable nodes cover all the user’s authorized cells. Hence, the completeness requirement is satisfied. \square

THEOREM 2. *The proposed schemes satisfy the soundness requirement.*

PROOF. It was proved that [4], under the graph-based key derivation framework, no coalition of users can recover a key (or distinguish a random key from the actual key) outside of

the nodes reachable from their assigned nodes. The security of the framework is based on the security of pseudo-random functions (secure symmetric encryption schemes are also required if the key indistinguishability requirement needs to be satisfied). Since our graph construction algorithms already guarantee that the reachable nodes cover no more than their authorized cells, the soundness requirement is satisfied. \square

7. CONCLUSION AND FUTURE WORK

In a practical sense the results we give in this paper put to rest the orthogonal range case because, although they are off from optimal space by a factor of $\log^* N$, that factor is practically always less than or equal 5 (as long as $N \leq 2^{65,536}$, which in practice means always). We believe it will be hard to get rid of the $\log^* N$ without an increase of more than 5 in the constant factor hiding behind the “big oh” notation. Future work will extend our scheme to arbitrarily shaped authorized regions, possibly with holes to capture the necessity of excluding sensitive geographic regions (e.g., military or critical infrastructures).

8. REFERENCES

- [1] S. Akl and P. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Transactions on Computer Systems*, 1(3):239–248, Sept. 1983.
- [2] C. Ardagna, M. Cremonini, E. Damiani, S. De Capitani di Vimercati, and P. Samarati. Supporting location-based conditions in access control policies. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS'06)*, pages 212–222, 2006.
- [3] M. Atallah, M. Blanton, and K. Frikken. Key management for non-tree access hierarchies. In *ACM Symposium on Access Control Models and Technologies (SACMAT'06)*. Full version available as *Technical Report TR 2007-30, CERIAS, Purdue University*.
- [4] M. J. Atallah, M. Blanton, and K. B. Frikken. Dynamic and efficient key management for access hierarchies. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (Alexandria, VA, USA, November 07 - 11, 2005)*. CCS '05. ACM, New York, NY, 190-202. The full version of the paper appears in *ACM Transactions on Information and System Security (TISSEC)*, v.12 n.3, p.1-43, January 2009.
- [5] M. J. Atallah, M. Blanton, and K. B. Frikken. Efficient techniques for realizing geo-spatial access control. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 82–92, New York, NY, USA, 2007. ACM.
- [6] M. J. Atallah, M. Blanton, and K. B. Frikken. Incorporating temporal capabilities in existing key management schemes. In J. Biskup and J. Lopez, editors, *Computer Security - ESORICS 2007, 12th European Symposium On Research In Computer Security, Dresden, Germany, September 24-26, 2007*, volume 4734 of *Lecture Notes in Computer Science*, pages 515–530. Springer, 2007.
- [7] M. J. Atallah, Y. Cho, and A. Kundu. Efficient data authentication in an environment of untrusted third-party distributors. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, México*, pages 696–704, 2008.
- [8] G. Ateniese, A. De Santis, A. Ferrara, and B. Masucci. Provably-secure time-bound hierarchical key assignment schemes. In *ACM Conference on Computer and Communications Security (CCS'06)*, 2006. Full version is available as Cryptology ePrint Archive Report 2006/255, <http://eprint.iacr.org/2006/255>.
- [9] V. Atluri and S. A. Chun. An authorization model for geospatial data. *IEEE Transactions on Dependable and Secure Computing*, 01(4):238–254, 2004.
- [10] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology - CRYPTO'96*, volume 1109, 1996.
- [11] E. Bertino, B. Catania, M. L. Damiani, and P. Perlasca. Geo-rbac: a spatially aware rbac. In E. Ferrari and G.-J. Ahn, editors, *Proceedings of SACMAT 2005, 10th ACM Symposium on Access Control Models and Technologies, Stockholm, Sweden, June 1-3, 2005*, pages 29–37. ACM, 2005.
- [12] B. Chazelle and B. Rosenberg. Computing partial sums in multidimensional arrays. In *SCG '89: Proceedings of the fifth annual symposium on Computational geometry*, pages 131–139, New York, NY, USA, 1989. ACM.
- [13] H. Chen, X. Ma, W. Hsu, N. Li, and Q. Wang. Access control friendly query verification for outsourced data publishing. In *ESORICS '08: Proceedings of the 13th European Symposium on Research in Computer Security*, pages 177–191, Berlin, Heidelberg, 2008. Springer-Verlag.
- [14] T. Chen, Y. Chung, and C. Tian. A novel key management scheme for dynamic access control in a user hierarchy. In *IEEE Annual International Computer Software and Applications Conference (COMPSAC'04)*, pages 396–401, Sept. 2004.
- [15] W. Cheng, H. Pang, and K.-L. Tan. Authenticating multi-dimensional query results in data publishing. In *Proceedings of Data and Applications Security XX, 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 60–73, 2006.
- [16] W. Cheng and K.-L. Tan. Authenticating knn query results in data publishing. In *Proceedings of Secure Data Management, 4th VLDB Workshop, SDM 2007*, pages 47–63, 2007.
- [17] H. Chien. Efficient time-bound hierarchical key assignment scheme. *IEEE Transactions of Knowledge and Data Engineering (TKDE)*, 16(10):1301–1304, 2004.
- [18] H. Chien and J. Jan. New hierarchical assignment without public key cryptography. *Computers & Security*, 22(6):523–526, 2003.
- [19] E. F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP(On-line Analytical Processing) to User-Analysts: An IT Mandate. In *Technical Report, Codd E.F. & Associates; 1993*.
- [20] J. Crampton, K. Martin, and P. Wild. On key assignment for hierarchical access control. In *IEEE Computer Security Foundations Workshop*

- (CSFW'06), 2006.
- [21] A. De Santis, A. Ferrara, and B. Masucci. Enforcing the security of a time-bound hierarchical key assignment scheme. *Information Sciences*, 176(12):1684–1694, 2006.
- [22] P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine. Authentic data publication over the internet. *J. Comput. Secur.*, 11(3):291–314, 2003.
- [23] M. T. Goodrich, R. Tamassia, and N. Triandopoulos. Super-efficient verification of dynamic outsourced databases. In T. Malkin, editor, *CT-RSA*, volume 4964 of *Lecture Notes in Computer Science*, pages 407–424. Springer, 2008.
- [24] S. Haber, W. Horne, T. Sander, and D. Yao. Privacy-preserving verification of aggregate queries on outsourced databases. Technical Report of HP Labs, HPL-2006-128.
- [25] H. Hacgumus, B. Iyer, and S. Mehrotra. Providing database as a service. In *Proceedings of the 18th International Conference on Data Engineering, 26 February - 1 March 2002, San Jose, CA*, 2002.
- [26] D. Harel and R. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal of Computing*, 13(2):338–355, 1984.
- [27] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 720–731. VLDB Endowment, 2004.
- [28] H. Huang and C. Chang. A new cryptographic key assignment scheme with time-constraint access control in a hierarchy. *Computer Standards & Interfaces*, 26:159–166, 2004.
- [29] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin. Dynamic authenticated index structures for outsourced databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29*, pages 121–132, 2006.
- [30] C. Lin. Hierarchical key assignment without public-key cryptography. *Computers & Security*, 20(7):612–619, 2001.
- [31] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced databases. In *Proceedings of ISOC Symposium on Network and Distributed Systems Security (NDSS'04)*, 2004.
- [32] M. Narasimha and G. Tsudik. DSAC: integrity for outsourced databases with signature aggregation and chaining. In *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, pages 235–236, 2005.
- [33] H. Pang, A. Jain, K. Ramamritham, and K.-L. Tan. Verifying completeness of relational query results in data publishing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, pages 407–418, 2005.
- [34] C. A. Patterson, R. R. Muntz, and C. M. Pancake. Challenges in location-aware computing. *IEEE Pervasive Computing*, 2(2):80–89, 2003.
- [35] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.
- [36] A. D. Santis, A. Ferrara, and B. Masucci. New constructions for provably-secure time-bound hierarchical key assignment schemes. In *ACM Symposium on Access Control Models and Technologies (SACMAT'07)*, 2007.
- [37] A. D. Santis, A. L. Ferrara, and B. Masucci. Efficient provably-secure hierarchical key assignment schemes. In L. Kucera and A. Kucera, editors, *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Český Krumlov, Czech Republic, August 26-31, 2007, Proceedings*, volume 4708 of *Lecture Notes in Computer Science*, pages 371–382. Springer, 2007.
- [38] E. Shi, J. Bethencourt, T.-H. H. Chan, D. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 350–364, Washington, DC, USA, 2007. IEEE Computer Society.
- [39] M. Srivatsa, A. Iyengar, J. Yin, and L. Liu. A scalable method for access control in location-based broadcast services. In *Proceedings of INFOCOM 2007. 26th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, pages 256–260, 2008.
- [40] Q. Tang and C. Mitchell. Comments on a cryptographic key assignment scheme for access control in a hierarchy. *Computer Standards & Interfaces*, 27:323–326, 2005.
- [41] W. Tzeng. A time-bound cryptographic key assignment scheme for access control in a hierarchy. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 14(1):182–188, 2002.
- [42] W. Tzeng. A secure system for data access based on anonymous authentication and time-dependent hierarchical keys. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS'06)*, pages 223–230, 2006.
- [43] S.-Y. Wang and C.-S. Lai. Merging: an efficient solution for a time-bound hierarchical key assignment scheme. *IEEE Transactions on Dependable and Secure Computing*, 3(1):91–100, 2006.
- [44] L. Xiong, S. Chitti, and L. Liu. Preserving data privacy in outsourcing data aggregation services. *ACM Trans. Internet Technol.*, 7(3):17, 2007.
- [45] A. C. Yao. Space-time tradeoff for answering range queries (extended abstract). In *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 128–136, New York, NY, USA, 1982. ACM.
- [46] J. Yeh. An RSA-based time-bound hierarchical key assignment scheme for electronic article subscription. In *ACM International Conference on Information and Knowledge Management (CIKM'05)*, pages 285–286, 2005.
- [47] X. Yi. Security of Chien's efficient time-bound hierarchical key assignment scheme. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(9):1298–1299, 2005.

- [48] X. Yi and Y. Ye. Security of Tzeng's time-bound key assignment scheme for access control in a hierarchy. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 15(4):1054–1055, 2003.
- [49] S. Zhong. A practical key management scheme for access control in a user hierarchy. *Computers & Security*, 21(8):750–759, 2002.