Scalable Key Management Algorithms for Location-Based Services

Mudhakar Srivatsa, Arun Iyengar, Jian Yin, and Ling Liu

***Abstract*—Secure media broadcast over the Internet poses unique security challenges. One important problem for public broadcast location-based services (LBS) is to enforce access control on a large number of subscribers. In such a system, a user typically subscribes to an LBS for a *time interval* (a, b) and a *spatial region* $(x_{bl}, y_{bl}, x_{\mathrm{tr}}, y_{\mathrm{tr}})$ according to a 3-dimensional spatial-temporal authorization model. In this paper, we argue that current approaches to access control using key management protocols are not scalable. Our proposal, `STauth`, minimizes the number of keys that needs to be distributed and is thus scalable to a large number of subscribers and the dimensionality of the authorization model. We also demonstrate applications of our algorithm to quantified-temporal access control (using $\forall$ and $\exists$ quantifications) and partial-order tree-based authorization models. We describe two implementations of our key management protocols on two diverse platforms: a broadcast service operating on top of a publish/subscribe infrastructure and an extension to the Google Maps API to support quality (resolution)-based access control. We analytically and experimentally show the performance and scalability benefits of our approach over traditional key management approaches.**

***Index Terms*—Access control, key management, location-based services (LBS), scalability and performance.**

I. Introduction

THE UBIQUITOUS nature of the Internet has resulted in widespread growth and deployment of location-based services (LBS) [2], [4], [5]. LBS (as the name indicates) provide information with spatial-temporal validity to potentially resource-constrained wireless and mobile subscribers. Example services include: 1) list all Italian restaurants in midtown Atlanta, 2) current traffic conditions at the junction of `peach tree parkway` and `peach tree circle`, 3) cheapest gas station in downtown Atlanta today. Secure LBS over an open channel such as the Internet or a wireless broadcast medium poses unique security challenges. LBS typically use a payment-based subscription model using 3-dimensional spatial-temporal authorization as follows: A paying user u subscribes for a spatial bounding box $(x_{bl}, y_{bl}, x_{\mathrm{tr}}, y_{\mathrm{tr}})$ and a time interval (a, b); the subscription fee may be an arbitrary function, say fee $\propto (x_{\mathrm{tr}} - x_{bl}) \times (y_{\mathrm{tr}} - y_{bl}) \times (b - a)$. A user u is allowed to read a broadcast from the LBS about a spatial coordinate (x, y) at time t if and only if $x_{bl} \leq x \leq x_{\mathrm{tr}}$, $y_{bl} \leq y \leq y_{\mathrm{tr}}$, and $a \leq t \leq b$.

A common solution for enforcing fine-grained access in such services is to encrypt the data and distribute the secret decryption key (group key) only to the legitimate receivers. The general approach is to use a key distribution center (KDC) for group key management. A group is defined as a set of users that hold equivalent authorizations. A user may be a part of zero (unauthorized user) or more groups. Group key management is complicated due to two reasons. 1) Group dynamics (a well-studied problem in literature) because of users joining and leaving a group at any time. Scalable algorithm to manage *one* group is well studied in literature: GKMP [21], LKH [31], [20], ELK [26]. These algorithms provide optimized solutions for a KDC to update the group key on member join and leave (subscription termination) events to ensure that a user is able to decrypt the data only when it is a member of the group of authorized users. 2) Large number of groups (new problem specific to LBS-like services). Using a spatial-temporal authorization model, each unit of data broadcast by an LBS may be destined to a potentially different set of subscribers. Hence, the number of such sets of subscribers (groups) may, in the worst case, be exponential (power set) in the number of subscribers. This largely limits the scalability of traditional group key management protocols in the context of LBS.

In this paper, we propose `STauth`, a secure, scalable, and efficient key management protocol for LBS-like services. `STauth` minimizes the number of keys that needs to be distributed and is thus scalable to a much higher number of subscribers and the dimensionality of the authorization model. We use N to denote the number of active users in the system and d to denote the dimensionality of an authorization model (for instance, the spatial-temporal authorization model discussed above is 3-dimensional $\langle x, y, t \rangle$).

In a group key management-based approach, one would define the set of users within a d-dimensional bounding box as a group. Suppose a user u_1 subscribes for a $d = 1$ spatial range (20, 30); then, we have one group $G = \{u_1\}$. Let us suppose that a new user u_2 subscribes for a range (25, 40); then, we have three groups: $G_1 = \{u_1\}$ [for the range (20, 25)], $G_2 = \{u_1, u_2\}$ [for the range (25, 30)], and $G_3 = \{u_2\}$ [for the range (30, 40)]. Observe that the group key management server has to not only maintain more keys (computing and storage cost) as the number of subscribers N increases, but also update keys at one or more

Manuscript received February 12, 2008; revised August 15, 2008; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor B. Levine. First published July 14, 2009; current version published October 14, 2009. The work of L. Liu was supported in part by grants from the NSF CyberTrust program, AFOSR, Intel, and an IBM faculty award. A preliminary version of this paper appears in IEEE INFOCOM, April 13–18, 2008, Phoenix, AZ: http://www.research.ibm.com/people/i/iyengar/Infocom08.pdf.

M. Srivatsa, A. Iyengar, and J. Yin are with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: msrivats@us.ibm.com; aruni@us.ibm.com; jianyin@us.ibm.com).

L. Liu is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: lingliu@cc.gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2008.2010222

existing subscribers as new users join/leave the network. Below, we briefly summarize the drawbacks of using existing key management protocols for location based services.

1) In the worst case, KDC manages $O(2^{dN})$ groups.
2) User join and leave requires the KDC to broadcast $O(2^{2d} * N)$ key update message.
3) The ELK protocol tolerates a certain level of packet losses during key updates; however, none of the protocols can tolerate arbitrary large packet losses.
4) Updates to the state maintained by the KDC (key hierarchy in LKH and ELK) have to be serialized, thereby making it hard to replicate the KDC on multiple servers. This makes it difficult to handle bursty loads on the KDC.
5) These protocols are vulnerable to purported *future* group-keys-based denial-of-service (DoS) attacks from unauthorized users (details follow in Section VI).
6) As described above, an authorized user buffers packets until it receives future group keys. This may cause large delays and jitters in actually decrypting and delivering the plain-text broadcast data to the client, thereby making this approach unsuitable for low-latency real-time broadcast services (like live audio/video teleconference). Packet losses during key updates and the DoS attack described above further complicate this problem.

Under the multidimensional authorization model, we use a simple yet powerful key management protocol using hierarchical key graphs [7], [12] with several features:

1) Number of groups managed by KDC is $O(1)$.
2) User join and leave cost is independent of N.
3) Requires no key update messages and is thus trivially resilient to arbitrary packet losses in key updates.
4) Allows the KDC to have a small, constant and stateless storage that is independent of N and d.
5) Allows dynamic and on-demand replication of KDC servers without requiring any interaction between the replicas (no concurrency control for serializing updates on KDC state).
6) Resilient to purported future group-key-based DoS attacks from unauthorized users.
7) Incurs only a small and constant (no jitter) computational overhead and is thus suitable even for low-latency real-time broadcast services.

In the rest of this paper, we first describe a scalable key management algorithm for temporal access control (Section II). We compare our algorithm analytically against other key management algorithms and show that our approach offers significant performance and scalability benefits. We demonstrate four applications of our algorithm. First, in Section III, we extend our algorithm to operate on quantification operators like $\forall$ and $\exists$ and demonstrate its usefulness to quantified-temporal access control. Second, in Section IV, we describe extensions to handle multidimensional authorization models (e.g.: spatial-temporal access control). Third, in Section V, we present constructions to support partial-order-trees-based authorization models (e.g.: spatial-quality access control). We sketch a prototype implementation of our proposal on a publish/subscribe broadcast service and evaluate its performance and scalability against traditional group key management approaches and more recent proposals in key management algorithms for geospatial access control ([7]–[9]). We also describe a prototype implementation of spatial-quality access control using the Google Maps API that demonstrates ease of use and deployability of our approach.

II. Temporal Authorization

A. Overview

In this section, we present techniques for handling temporal authorizations (one-dimensional) in broadcast services. In this scenario, we assume that a user needs to subscribe (by paying a fee) to access the broadcast service. Each subscription has a lifetime indicated by a time interval (a, b); note that (a, b) could be different and highly fine-grained for different user subscriptions. When a user subscribes for a broadcast service S from time (a, b), the service provider issues an authorization key $K^{a,b}$ to the user u. This ensures that:

- Given $K^{a,b}$, a user u can efficiently derive $K^{t,t}$ if $a \leq t \leq b$.
- Given $K^{a,b}$, it is computationally infeasible for a user u to guess $K^{t,t}$ if $t < a$ or $t > b$.

The primitive described above helps us to construct a very simple and efficient protocol for temporal access control on broadcast services. At any given time instant t, the service provider broadcasts a packet P (of, say, audio/video data) as follows:

- Get current time instant t and compute $K^{t,t}$.
- Broadcast $\langle t, E_{K^{t,t}}(P), \mathrm{MAC}_{K^{t,t}}(P)\rangle$.

$E_K(x)$ and $\mathrm{MAC}_K(x)$ denote an encryption and a message authentication code of a string x, respectively. Note that all users can potentially receive the broadcast message. An authorized subscriber decrypts the payload P as follows:

- Receive the broadcast message $\langle t, E_{K^{t,t}}(P), \mathrm{MAC}_{K^{t,t}}(P)\rangle$. Note that the time instant t is in plain-text.
- A subscriber is authorized if it has a temporal authorization for some time period (a, b) such that $a \leq t \leq b$. An authorized subscriber can compute the decryption key $K^{t,t}$ from $K^{a,b}$, decrypts the broadcast message to obtain the payload P, and checks its integrity.

The property of the authorization key $K^{a,b}$ ensures that one can efficiently compute $K^{t,t}$ from $K^{a,b}$ if and only if $a \leq t \leq b$. In the following section, we present an algorithm to efficiently and securely construct such keys using hierarchical key graphs.

B. Key Management Algorithm

In this section, we describe techniques to construct keys using hierarchical key graphs [7], [12], [31] that satisfy the primitive described in Section II-A. We first introduce some notation and parameters used in our algorithm (see Table I). Let $(0, T_{\max})$ denote the time horizon of interest. Let δt s denote the smallest time granularity of interest. Let time equal to t denote the tth time unit, where one unit $\mathrm{time} = \delta t$ s. Our algorithms efficiently support temporal authorization at very low granularities ($\delta t \sim 10^{-3}$ or 10^{-6}). We associate a key $K^{a,b}(S)$ as the authorization key that permits a user u to access a broadcast service S in the time interval (a, b).

We now construct a key tree that satisfies the property that a user u can efficiently guess $K^{t,t}$ from $K^{a,b}$ if and only if $a \leq t \leq b$. Each element in the key tree is labeled with a time interval starting with the root $(0, T_{\max})$. Each element (a, b) in

TABLE I
NOTATION

N	number of users
H	PRF
X	xor operation
E	encryption function
D	decryption function
K	key size in bits
n_1, n_2	ELK parameters
T_{max}	total time period
$rate$	message broadcast rate
δt	time granularity

the key tree has two children labeled with time intervals $(a, \frac{a+b}{2})$ and $(\frac{a+b}{2}+1, b)$. We associate a key $K^{a,b}(S)$ with every element (a, b) in the key tree. The keys associated with the elements of the key tree are derived recursively as follows:

$$K^{a,\frac{a+b}{2}}(S) = H(K^{a,b}(S), 0)$$
$$K^{\frac{a+b}{2}+1,b}(S) = H(K^{a,b}(S), 1) \quad (1)$$

where $H(K, x)$ denotes output of a pseudo-random function (PRF) keyed by K for which the range is sufficiently large that the probability of collision is negligible. The root of the key tree has a key computed using the KDC's secret master key MK, and S is the name of the broadcast service $K^{0,T_{\max}}(S) = H(\mathrm{MK}, S)$. Observe that given $K^{a,b}(S)$, one can derive all keys $\{K^{t,t}(S) : a \leq t \leq b\}$. Also, deriving the key $K^{t,t}(S)$ for any $a \leq t \leq b$ from $K^{a,b}(S)$ requires no more than $\log_2 \frac{b-a}{\delta t}$ applications of H. Algorithm 1 shows an algorithm for deriving $K^{t,t}$ from $K^{a,b}$.

Algorithm 1: Key Derivation

Input: $t, K^{a,b}$
Output: $K^{t,t}$
DERIVE$(t, K^{a,b})$
1) **if** $t < a$ or $t > b$
2) **return** $\perp$
3) $\mathrm{mid} \leftarrow \frac{a+b}{2}$
4) **if** $t = \mathrm{mid}$
5) **return** $K^{a,b}$
6) **if** $t < \mathrm{mid}$
7) $K^{a,\mathrm{mid}} \leftarrow H(K^{a,b}, 0)$
8) **return** DERIVE$(t, K^{a,\mathrm{mid}})$
9) **else**
10) $K^{\mathrm{mid}+1,b} \leftarrow H(K^{a,b}, 1)$
11) **return** DERIVE$(t, K^{\mathrm{mid}+1,b})$

Fig. 1 illustrates the construction of our key tree assuming $T_{\max} = 31$ time units. We derive $K^{0,31}(S) = H(\mathrm{MK}, S)$. Then, we compute $K^{0,15}(S) = H(K^{0,31}(S), 0)$ and $K^{16,31}(S) = H(K^{0,31}(S), 1)$. One can recursively extend this definition to any arbitrarily small time granularity at the expense of additional key derivation cost.

Having described the construction of our key tree, we pick an authorization key for any arbitrary time interval (a, b) as follows. One can show that any time interval (a, b) can be *partitioned* into no more than $2\log_2 \frac{T_{\max}}{\delta t} - 2$ elements in the key tree. For example, given a time interval (8, 19), we partition the time interval into two subintervals (8, 15) and (16, 19) (see Fig. 1).

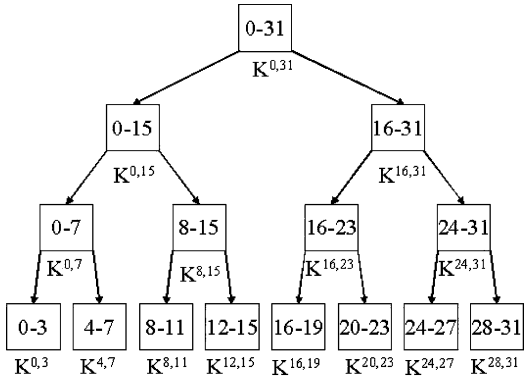

Fig. 1. Authorization key tree.

We provide temporal authorization for a time interval (8, 19) by issuing two authorization keys $K^{8,15}(S)$ and $K^{16,19}(S)$.

Security Analysis. We present a security analysis of our protocol using the following cryptographic game STauth:
Setup: The KDC generates a random ρ-bit private master key MK and outputs a public security parameter ρ and a PRF $H : \{0,1\}^\rho \times \{0,1\}^* \rightarrow \{0,1\}^\rho$.
Query: Subscriber adaptively issues $q = \mathrm{poly}(\rho)$ queries to the KDC for time intervals (a_i, b_i) $[0 \leq i < q$ and $(a_i, b_i) \neq (0, T_{\max})]$. The KDC returns K^{a_i,b_i} for the ith query.
Challenge: Subscriber picks $t \notin (a_i, b_i)$ (for any $0 \leq i < q$). The KDC returns a random permutation of the set $\{K^{t,t}, R\}$, where R is a random bit string of length ρ. The KDC challenges the subscriber to distinguish between $K^{t,t}$ and R in the output.

Let $\mathrm{dist}(K, R)$ denote the probability with which a probabilistic poly time (PPT) subscriber can distinguish key K from a random bit string R (of equal lengths). We note that for any $(a, b) = \mathrm{anc}(a_i, b_i)$, where anc denotes an ancestor of the node (a_i, b_i) in the authorization key tree, the subscriber can easily distinguish $K^{a,b}$ from R (by deriving K^{a_i,b_i} from $K^{a,b}$); hence, $\mathrm{dist}(K^{a,b}, R) = 1$. Trivially, given K^{a_i,b_i} for any range (a_i, b_i), $\mathrm{dist}(K^{0,T_{\max}}, R) = 1$. We hypothesize that any $K^{a,b}$ (where (a, b) is an ancestor of (a_i, b_i)) is secure against key recovery, while it fails to satisfy key indistinguishability [18]. However, we note that only the leaf nodes in the authorization key tree (namely, $K^{t,t}$) are used for encrypting broadcast messages. Hence, the challenge phase attempts to establish key indistinguishability only for those encryption keys (note that composability with secure encryption algorithm requires that the encryption keys satisfy key indistinguishability). Fig. 2 shows keys that are resistant to key recovery (KR) and key indistinguishability (KI) when $K^{0,7}$ is revealed to the subscriber.

The advantage for a subscriber $\mathrm{Adv}_{\mathrm{STauth}}$ is defined as $\mathrm{dist}(K^{t,t}, R)$. Let $\mathrm{Adv}_{\mathrm{PRF}}$ denote the advantage for an subscriber against a pseudo-random function H defined using the following cryptographic game PRF.
Setup: The KDC generates a private ρ-bit key K and outputs a public security parameter ρ and a PRF $H : \{0,1\}^\rho \times \{0,1\}^* \rightarrow \{0,1\}^\rho$.
Query: Subscriber adaptively issues $q = \mathrm{poly}(\rho)$ queries to the KDC for inputs $x_0, x_2, \ldots, x_{q-1}$. The KDC returns $y_i = H(K, x_i)$ for the ith query.
Challenge: The subscriber picks $x \notin \{x_i\}$, and the KDC returns a random permutation of the set $\{y, R\}$ such that $y = H(K, x)$, where R is a random bit string. The KDC

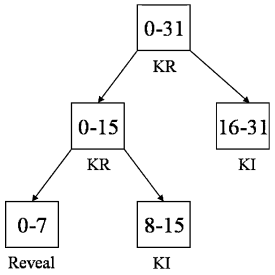

Fig. 2. Key recovery and key indistinguishability—Set of revealed keys: $\{K^{0,7}\}$, keys resistant to recovery $\mathrm{KR} = \{K^{0,15}, K^{0,31}\}$ and keys that are indistinguishable from random $\mathrm{KI} = \{K^{8,15}, K^{16,31}, \ldots\}$.

challenges the subscriber to distinguish between y and R in the output.

Theorem 1: For any PPT adversary, $\mathrm{Adv}_{\mathrm{STauth}} \leq T_{\max} * \mathrm{Adv}_{\mathrm{PRF}}$, where $T_{\max}$ is the size of the temporal dimension.

Proof: Let A denote a PPT algorithm that distinguishes $K^{t,t}$ and R with probability $\mathrm{Adv}_{\mathrm{STauth}}$. Let us consider a simple case with $T_{\max} = 2$. We have three keys $K^{0,1}, K^{0,0} = H(K^{0,1}, 0)$, and $K^{1,1} = H(K^{0,1}, 1)$. Let us suppose in the query phase of `STauth` game, the subscriber queries for $K^{0,0}$. In the challenge phase, the subscriber picks $t = 1$ and is presented with a random permutation of the set $\{K^{1,1}, R\}$. It is easy to see that if the subscriber can use algorithm A to distinguish $K^{1,1}$ from R with probability $\mathrm{Adv}_{\mathrm{STauth}}$, then it can defeat PRF game with at least the same probability, namely $\mathrm{Adv}_{\mathrm{PRF}} \geq \mathrm{Adv}_{\mathrm{STauth}}$. One can use proof techniques similar to [18] to show that $\mathrm{Adv}_{\mathrm{STauth}}$ is no more than $\mathrm{Adv}_{\mathrm{PRF}}$ amplified by the maximum number of keys queried in the `STauth` game $(T_{\max})$. The proof details are outside the scope of this paper. ■

Cost Analysis. In general, if one uses a r-ary key tree $(r \geq 2)$, any range can always be subdivided into no more than $r(\log_r(\frac{T_{\max}}{\delta t}) - 1)$ subintervals. One can show that this is a monotonically increasing function in r (for $r \geq 2$) and thus has a minimum value when $r = 2$. One can also show that if the time interval (a, b) were chosen uniformly and randomly from $(0, T_{\max})$, then on average (a, b) can be subdivided into $(r-1)\log_r \frac{b-a}{\delta t}$ subintervals. This is also a monotonically increasing function in r (for $r \geq 2$) and thus has a minimum value at $r = 2$. However, as r increases, the height of the key tree $(\log_r(\frac{T_{\max}}{\delta t}))$ decreases;, that is, the cost of key derivation decreases monotonically with r. However, since the PRF H is computationally inexpensive (<1 μs on a typical 900 MHz Pentium III processor), we focus our efforts on minimizing the size of the authorization key rather than the key derivation cost. Tables II and III show the maximum and the average number of keys and computation time required for different values of δt for a time interval of one year using a binary authorization key tree $(r = 2)$, respectively.

C. Comparison With Other Approaches

In this section, we present an analytical comparison of our approach against other group key management protocols. `Simple` uses a key $K(u)$ for a user u. When the group key needs to be updated (because of some user joining or leaving the system), the KDC chooses a new random group key. The KDC sends one message per group member u that includes the new group key encrypted with $K(u)$. `LKH` [31] builds a logical key hierarchy on the set of authorized users to enhance the efficiency of the key update protocol. `ELK` [26] introduces the concepts of hints to enhance the efficiency of LKH protocol and improve its resilience to arbitrary packet loss of key update messages.

TABLE II
MAXIMUM NUMBER OF KEYS AND COMPUTATION TIME

δt	Num Keys	Time (μs)
one month	6	12.74
one week	10	20.02
one day	16	30.94
one hour	26	49.14
one minute	38	70.98
one second	48	89.18
one millisec	68	125.58

TABLE III
AVERAGE NUMBER OF KEYS AND COMPUTATION TIME WITH $\delta t = 1$ s

$b - a$	Num Keys	Time (μs)
one month	21	40.04
one week	19	38.22
one day	16	35.49
one hour	11	30.94
one minute	5	25.48
one second	1	21.84

Atallah *et al.* [7]–[9], (henceforth referred to as `TAC` in this paper) have proposed key management algorithms for handling temporal capabilities. Their approach presents an alternate implementation of our high-level protocol described in Section II-A. Similar to our approach, they use a directed acyclic graph (DAG) over the 1-dimensional space (e.g.: time). The atomic primitive supported by their approach is to derive a key along a directed edge from a node with label l_u to a node with label l_v. Each node v in the graph is associated with a key K_v; the keys K_v are generated randomly for every node v. Given a directed edge, $l_u \rightarrow l_v$ is labeled with a public information $y_{u,v} = K_v \oplus F_{K_u}(l_v)$, where $F_K(s)$ denotes a family of pseudo-random functions on an input key K and string s. Given K_u and the public label $y_{u,v}, K_v$ is derived as $K_v = F_{K_u}(l_v) \oplus y_{u,v}$. The authors propose using short cut edges to trade off the size of public storage and the key derivation cost.

On the positive side, `TAC` requires only $O(1)$ keys to be distributed when a new user joins the network, while our approach requires $O(\log T)$ keys. We note that this is a one-time communication cost incurred when a user subscribes to the system. `TAC` incurs $O(1)$ key derivation cost, in comparison to $O(\log T)$ key derivation cost incurred by our approach. We show below that one can reduce the amortized key derivation cost to $O(1)$ in our approach using a key-caching-based approach.

On the flip side, `TAC` incurs $O(1)$ communication cost for key derivation. While `TAC` does not have to communicate with the KDC to derive a key, it does require access to authenticated 'public information' (namely, labels on directed edges in `TAC`) in order to derive keys. This public information could be retrieved by a subscriber once-for-all when he or she joins the network or on an on-demand basis. In either approach, the amortized communication cost to pull out public information per derived key is O(1). In contrast, `STauth` requires no public information and, thus, no communication cost for key derivation.

TABLE IV
STORAGE COST

	KDC	user
Simple	$N * K$	K
LKH	$(2N-1)K$	$(\log_2 N+1)K$
ELK	$(2N-1)K$	$(\log_2 N+1)K$
TAC	$T_{max}\log\log T_{max} * K$	$3K$
STauth (max)	K	$(2\log_2 \frac{T_{max}}{\delta t}-2)K$
STauth (avg)	K	$\log_2 \frac{b-a}{\delta t} * K$

TAC requires at least $O(T*\log\log T)$ *public storage*. Using a fine-grained access control (say, $\delta t = 1$ s), $T_{\max}$ for one year is about $3.15*10^7$. Hence, the cost of public storage may become prohibitively high; on the other hand, our approach can support very fine granularity (say, $\delta t = 1\ \mu$s). While public storage may be made available to all users (authorized or not) without compromising access control, the integrity and availability of public storage must be guaranteed. For instance, the public storage may become a target for DoS attacks; also, a compromised public storage system may serve corrupted data, making it impossible for legitimate users to derive the decryption keys.

Security Properties. Table V compares the properties of different group key management approaches. The LKH and ELK approach have a centralized key graph data structure that is nontrivial to be distributed amongst multiple KDCs. On the other hand, our approach can use multiple KDC servers by just sharing the read-only master key MK amongst them. Note that since all temporal authorization keys are derivable from the master key MK we do not require the KDC servers to share and update a common data structure. This allows on-demand creation of KDC server replicas to handle bursty KDC traffic. Our approach does not require a key update protocol, thereby making it trivially tolerant to arbitrary packet losses in key update messages. Finally, our approach does not require a multicast channel between the KDC and the user, since the KDC does not have to broadcast any key update messages to the users.

Storage Cost. Table IV compares the storage cost at the KDC and the users for different approaches. Our approach requires the KDC to only store the master key MK (rest of the keys can be computed on the fly). On the other hand, in the LKH and the ELK approach, the storage cost at the KDC grows linearly with the number of users N. In our approach, the storage cost at a user is on an average logarithmic in the length of the subscription time interval.

Communication Cost. Table VI compares the communication cost at the KDC and the users for different key management protocols. The key advantage of our approach is that a key does not need to be updated once it is given to the user. A join operation requires only an interaction between the KDC and the new user; a subscription terminate operation is cost-free. One should note that the temporal authorization model simplifies the user leave operation by *a priori* determining the time interval (a, b). On the other hand, LKH join, LKH leave and ELK leave send $O(\log_2 N)$-size messages to all the users $O(N)$; ELK join sends an $O(\log_2 N)$-size message only to the new user while compromising backward secrecy for at most one *time interval*. Furthermore, the KDC has to maintain the set of active users in order to update the logical key hierarchy data structure.

Computation Cost. Table VII compares the computation cost at the KDC and the users for different approaches. Our approach requires only simple PRF computations at the KDC to handle a new user join. The LKH join, LKH leave and ELK leave need to encrypt and update at least $O(\log_2 N)$ keys in the key graph and broadcast a key update message to all the users. As described earlier, our approach has zero cost for key update and user leaves. However, our approach incurs a small computation cost for processing broadcast packets. Given the time instant t in the packet header, the user has to compute the key $K^{t,t}$ from an authorization key $K^{a,b}(a \le t \le b)$. This may require $\log_2 \frac{b-a}{\delta t}$ applications of H. Using standard cryptographic algorithms (say, HMAC-SHA [17], [23] for H and AES-CBC-128 [25] for E), the cost of key derivation will be about two orders of magnitude smaller than that of encryption/decryption, thereby making this approach suitable for low-latency real-time applications (like audio and video broadcast for a teleconference). On the other hand, low-latency real-time applications that use LKH and ELK may experience large delays and unexpected jitters due to key updates and packet losses during key updates (application packets need to be buffered until the user receives an updated key). Indeed, an unauthorized subscriber (adversary) may exploit this vulnerability to launch a DoS attack by flooding subscribers with applications packets that are purportedly encrypted with *future* group keys. We can easily mitigate such an attack in our approach by appending a message authentication code (MAC) $\mathrm{MAC}_{K^{t,t}}(P)$ to the broadcast message.

Key Caching. One can additionally use a caching mechanism described below to decrease the key derivation cost. Let us suppose that a user received a broadcast packet P at time t. In the process of computing $K^{t,t}$ from its authorization key $K^{a,b}(a \le t \le b)$, the user computes several intermediate keys $K^{a',b'}(a \le a' \le t \le b' \le b)$. The user can cache these intermediate keys for future use. Say the user were to receive its next broadcast packet P' at time t'; then, the user could potentially compute $K^{t',t'}$ from some $K^{a',b'}$ such that $a \le a' \le t \le t' \le b' \le b$. Indeed, this would require only $\log_2 \frac{b'-a'}{\delta t}$ applications of $H(b'-a' \le b-a)$. One can show that if the mean interpacket arrival time is $\frac{1}{\mathrm{rate}}$, then the mean per-packet key derivation cost drops to $-H\log_2(\mathrm{rate}*\delta t)$ (assuming, $\delta t < \frac{1}{\mathrm{rate}}$).

III. QUANTIFICATIONS

A. Overview

In this section, we present an application of our key management algorithm to handle universal and existential quantifications over the temporal domain. We motivate our algorithm using quantified-temporal access control on broadcast data. Informally, a $\forall$-temporal access control constraint is specified using a three tuple: $(\forall, \mathrm{beg}, \mathrm{end})$. A user u satisfies this constraint if its temporal authorization holds *for all* time instants $t \in (\mathrm{beg}, \mathrm{end})$. Similarly, a user u satisfies a $\exists$-temporal access control rule $(\exists, \mathrm{beg}, \mathrm{end})$ if its temporal authorizations hold *for some* time instant $t \in (\mathrm{beg}, \mathrm{end})$.

In the context of broadcast services, we assume that every unit of broadcast data (say, an object o or a file f) is tagged with a quantified-temporal access control rule. We also assume that the broadcast data is encrypted with a randomly chosen secret key rand_K. Now, we require the encrypted broadcast data be made publicly available to all subscribers. However, the data should

TABLE V
SECURITY PROPERTIES

	Forward/Backward Secrecy	Collusion Resistance	Distributed KDC	KDC-User Channel	Reliable Key Update
Simple	Yes	Yes	Yes	unicast	No
LKH	Yes	Yes	No	multicast	No
ELK	Yes	Yes	No	multicast	Yes
TAC	Yes	Yes	Yes	unicast	Yes
STauth	Yes	Yes	Yes	unicast	Yes

TABLE VI
COMMUNICATION COST

	Join (KDC)	Join (users)	Terminate (KDC)	Terminate (users)	Msg (user)
Simple	$N * K$	$N * K$	$N * K$	$N * K$	-
LKH	$(\log_2 N + 1)K$	$(\log_2 N + 1) * N * K$	$2 \log_2 N * K$	$2 \log_2 N * N * K$	-
ELK	$(\log_2 N + 1)K$	$(\log_2 N + 1) * K$	$(\log_2 N - 1)(n_1 + n_2)$	$(\log_2 N - 1) * (n_1 + n_2) * N$	-
TAC	$3K$	$3K$	-	-	$5K$
STauth (max)	$(2 \log_2 \frac{T_{max}}{\delta t} - 2)K$	$(2 \log_2 \frac{T_{max}}{\delta t} - 2)K$	-	-	-
STauth (avg)	$\log_2 \frac{b-a}{\delta t} * K$	$\log_2 \frac{b-a}{\delta t} * K$	-	-	-

TABLE VII
COMPUTATION COST

	Join (KDC)	Join (users)	Terminate (KDC)	Terminate (users)	Msg (user)
Simple	$N * E$	$N * D$	$N * E$	$N * D$	D
LKH	$\log_2 N(H + 3E)$	$(\log_2 N + 1) * N * D$	$2 \log_2 N * E$	$\log_2 N * D$	D
ELK	$2(2N - 1)H + 2E + (\log_2 N + 1)E$	-	$8 \log_2 N * E$	$\log_2 N * D + 5 \log_2 N * E$	D
TAC	-	-	-	-	$5H + D$
STauth (max)	$(4 \log_2 \frac{T_{max}}{\delta t} - 2)H$	-	-	-	$H \log_2 \frac{b-a}{\delta t} + D$
STauth (avg)	$(\log_2 \frac{T_{max}}{\delta t} + \log_2 \frac{b-a}{\delta t} - 1)H$	-	-	-	$-H \log_2(rate * \delta t) + D$

be intelligible to a user only if its temporal authorization (a, b) satisfies the quantified-temporal constraint associated with the data. Observe that a user u with a temporal authorization (a, b) can satisfy a $(\forall, \text{beg}, \text{end})$ constraint if and only if $a \leq \text{beg} \leq \text{end} \leq b$ and satisfy a $(\exists, \text{beg}, \text{end})$ constraint if and only if $(b \geq \text{beg} \wedge a \leq \text{end})$.

In our key management algorithm, we associate an authorization key $\text{AK}^{a,b}$ with a time interval (a, b). We associate an encryption key $\text{EK}^{\forall,a,b}$ with a $\forall$-temporal constraint $(\forall, \text{beg}, \text{end})$ and $\text{EK}^{\exists,a,b}$ with a $\exists$-temporal constraint $(\exists, \text{beg}, \text{end})$. Our enforcement protocol is similar to Section II. A broadcast data with a quantified-access control constraint $(q, \text{beg}, \text{end})$ $(q \in \{\forall, \exists\})$ is encrypted with an encryption key $\text{EK}^{q,\text{beg},\text{end}}$. Only an authorized user can derive the encryption key $\text{EK}^{q,\text{beg},\text{end}}$ from its authorization key $\text{AK}^{a,b}$ and thus decrypt the broadcast data. We construct the authorization key $\text{AK}^{a,b}$ and the encryption keys $\text{EK}^{\forall,a,b}$ and $\text{EK}^{\exists,a,b}$ such that:

- Given an authorization key $\text{AK}^{a,b}$, a user u can efficiently derive any encryption key $\text{EK}^{\forall,\text{beg},\text{end}}$ if $a \leq \text{beg} \leq \text{end} \leq b$.
- Given an authorization key $\text{AK}^{a,b}$, it should be computationally infeasible to derive any encryption key $\text{EK}^{\forall,\text{beg},\text{end}}$ if $\text{beg} < a \vee \text{end} > b$.
- Given an authorization key $\text{AK}^{a,b}$, a user u can efficiently derive any encryption key $\text{EK}^{\exists,\text{beg},\text{end}}$ if $b \geq \text{beg} \wedge a \leq \text{end}$.
- Given an authorization key $\text{AK}^{a,b}$, it should be computationally infeasible to derive any encryption key $\text{EK}^{\exists,\text{beg},\text{end}}$ if $b < \text{beg} \vee a > \text{end}$.

B. Key Management Algorithm

1) $\forall$-Temporal Authorization: We observe that a $\forall$-temporal constraint reduces to that of a simple temporal access control constraint when $\text{beg} = \text{end} = t$ (see Section II). We leverage the same key management algorithm described in Section II as follows. Given a time interval (a, b), the authorization key $\text{AK}^{a,b} = K^{a,b}$ is constructed using the same key management algorithm as that described in Section II. Now, we generate the encryption key $\text{EK}^{\forall,\text{beg},\text{end}}$ as follows. Let $(\text{beg}_1, \text{end}_1), \ldots, (\text{beg}_n, \text{end}_n)$ minimally partition the range (beg, end), such that $(\text{beg}_i, \text{end}_i)$ (for all $1 \leq i \leq n$) are elements on the key tree. Now, we construct the encryption key as

$$\text{EK}^{\forall,\text{beg},\text{end}} = \bigoplus_{i=1}^{n} K^{\text{beg}_i,\text{end}_i}. \tag{2}$$

For example, $\text{EK}^{\forall,0,11} = K^{0,7} \oplus K^{8,11}$. Observe that given an authorization key $\text{AK}^{a,b}$ such that $a \leq \text{beg} \leq \text{end} \leq b$, an authorized user can efficiently derive $K^{\text{beg}_i,\text{end}_i}$ (for all $1 \leq i \leq n$) since $a \leq \text{beg} \leq \text{beg}_i \leq \text{end}_i \leq \text{end} \leq b$. For example, an authorized user u with authorization key $K^{0,15}$ can derive $K^{0,7} = H(K^{0,15}, 0), K^{8,11} = H(H(K^{0,15}, 1), 0)$ and encryption key $\text{EK}^{\forall,0,11} = K^{0,7} \oplus K^{8,11}$.

Let us consider an unauthorized user u'. A user u' is unauthorized if the temporal authorization for user u' fails for some time instant $t \in (\text{beg}, \text{end})$. From the temporal authorization model (Section II) it is evident that it should be computationally infeasible for the user u' to guess $K^{t,t}$. Hence, it should be computationally infeasible for the user u' to guess $K^{\text{beg}_j,\text{end}_j}$ such that $\text{beg}_j \leq t \leq \text{end}_j$ and $1 \leq j \leq n$. Note that such a j exists since $\text{beg} \leq t \leq \text{end}$ and $(\text{beg}_i, \text{end}_i)$ partitions the range (beg, end). Hence, without knowing $K^{\text{beg}_j,\text{end}_j}$, it is infeasible for the unauthorized user u' to guess the encryption key $\text{EK}^{\forall,\text{beg},\text{end}}$.

2) $\exists$-Temporal Authorization: We now focus on the $(\exists, \text{beg}, \text{end})$ access control constraints. We leverage the same key management algorithm described in Section II to handle

$\exists$-temporal constraints as follows. Let us suppose that a user u is authorized for some time interval (a, b). For the sake of simplicity, let us suppose that (a, b) exactly matches an element in the authorization key tree in Section II. If not, the algorithm described below should be duplicated for every partition of (a, b) in the authorization key tree. Let $(a_1, b_1), \ldots (a_m, b_m)$ denote the ancestors of element (a, b) on the authorization key tree with $(a_1, b_1) = (0, T_{\max})$. Now, the authorization key for time interval (a, b) is constructed as

$$\mathrm{AK}^{\exists,a,b} = K^{a,b}, F(K^{a_1,b_1}), \ldots, F(K^{a_m,b_m}). \quad (3)$$

For example, the authorization key for time interval (0, 15) is $\mathrm{AK}^{\exists,0,15} = \{K^{0,15}, F(K^{0,31})\}$. where F is a one-way collision-free hash function. The encryption key for a broadcast data with access control constraint $(\exists, \mathrm{beg}, \mathrm{end})$ is constructed as

$$\mathrm{EK}^{\exists,\mathrm{beg},\mathrm{end}} = F(K^{\mathrm{beg}_1,\mathrm{end}_1}), \ldots, F(K^{\mathrm{beg}_n,\mathrm{end}_n}). \quad (4)$$

Let $(\mathrm{beg}_1, \mathrm{end}_1), \ldots, (\mathrm{beg}_n, \mathrm{end}_n)$ minimally partition the range (beg, end) such that $(\mathrm{beg}_i, \mathrm{end}_i)$ (for all $1 \leq i \leq n$) are elements on the key tree. For example, the encryption keys for an access control constraint $(\exists, 0, 11)$ is $\mathrm{EK}^{\exists,0,11} = \{F(K^{0,7}), F(K^{8,11})\}$. The encryption key rand_K for the broadcast data is randomly chosen, and rand_K is encrypted using key encryption keys from $\mathrm{EK}^{\exists,\mathrm{beg},\mathrm{end}}$ and broadcast along with the data. An authorized user u with $\mathrm{AK}^{\exists,0,15} = \{K^{0,15}, F(K^{0,31})\}$ can compute $K^{0,7}$ from the authorization key $K^{0,15}$. It can then use $F(K^{0,7})$ to decrypt the file f's metadata and obtain the file encryption key $K(f)$.

One can easily observe that the authorization key $\mathrm{AK}^{\exists,a,b}$ satisfies the following property: Given any element (x, y) in the authorization tree such that $a \leq x \leq y \leq b$, a user u can compute $H(K^{\mathrm{anc}x,\mathrm{anc}y})$ for all ancestors $(\mathrm{anc}x, \mathrm{anc}y)$ of the element (x, y) on the authorization key tree. Recall that a user u satisfies the constraint $(\exists, \mathrm{beg}, \mathrm{end})$ if and only if there exists a time instant $t \in (\mathrm{beg}, \mathrm{end}) \cap (a, b)$. Since, $t \in (a, b)$, the user u can compute $F(K^{\mathrm{anct1},\mathrm{anct2})})$ for all ancestors (anct1, anct2) of the element (t, t). Since $t \in (\mathrm{beg}, \mathrm{end})$, there exists a partition j of (beg, end) such that $t \in (\mathrm{beg}_j, \mathrm{end}_j)$ $(1 \leq j \leq n)$; that is, $(\mathrm{beg}_j, \mathrm{end}_j)$ is an ancestor of the element (t, t) in the authorization tree. Hence, the user u can compute the encryption key $F(K^{\mathrm{beg}_j,\mathrm{end}_j})$.

Let us consider an unauthorized user u'. A user u' is unauthorized if $a > \mathrm{end} \vee b < \mathrm{beg}$. Let us consider the first case $a > \mathrm{end}$. Hence, for all partitions of $(\mathrm{beg}_1, \mathrm{end}_1), \cdots (\mathrm{beg}_n, \mathrm{end}_n)$ of the range $(\mathrm{beg}, \mathrm{end}), b \geq a > \mathrm{end}_i$. Therefore, for no $i, (\mathrm{beg}_i, \mathrm{end}_i) \in (a, b)$; that is, it is infeasible to guess $K^{\mathrm{beg}_i,\mathrm{end}_i}$ [and thus guess the encryption key $F(K^{\mathrm{beg}_i,\mathrm{end}_i})$] from $K^{a,b}$ (and thus from the authorization key $\mathrm{AK}^{\exists,a,b}$). Also, for no $i, (a, b) \in (\mathrm{beg}_i, \mathrm{end}_i)$; that is, the element $(\mathrm{beg}_i, \mathrm{end}_i)$ is not an ancestor of the element (a, b) on the authorization key tree, and thus the authorization key $\mathrm{AK}^{\exists,a,b}$ does not include the encryption key $F(K^{\mathrm{beg}_i,\mathrm{end}_i})$. A similar argument holds for the second case $b < \mathrm{beg}$.

C. Comparison With Group Key Management Approaches

We compare the cost of our key management algorithm with the group key management approaches. Tables VIII–X show the costs for our key management algorithm. The security properties of our approach are identical to that of Table V. Observe that the storage, communication, and computation costs are small and independent of the number of users in the system. Also, the key derivation cost is very small when compared to the decryption cost, thereby ensuring that our approach adds only a small $(\sim 1\%)$ overhead.

TABLE VIII
QUANTIFICATIONS: STORAGE COST

	KDC	user
$\forall$ (max)	K	$(2 \log_2 \frac{T_{max}}{\delta t} - 1)K$
$\forall$ (avg)	K	$(\log_2 \frac{b-a}{\delta t})K$
$\exists$ (max)	K	$(4 \log_2 \frac{T_{max}}{\delta t} - 3)K$
$\exists$ (avg)	K	$(2 * \log_2 \frac{b-a}{\delta t} + 1)K$

IV. MULTIDIMENSIONAL AUTHORIZATION

A. Overview

In this section, we extend our key management algorithms to operate on multidimensional authorization models. In this section, we use LBS as a motivating example. LBS provide information with spatial-temporal validity—say, traffic information at the junction (x, y) at time t. LBS use a spatial-temporal authorization model as follows: A user u subscribes for a spatial bounding box $(x_{bl}, y_{bl}, x_{\mathrm{tr}}, y_{\mathrm{tr}})$ and a time interval (a, b). A user u is allowed to read a broadcast from the LBS about a spatial coordinate (x, y) at time t if and only if $x_{bl} \leq x \leq x_{\mathrm{tr}}$, $y_{bl} \leq y \leq y_{\mathrm{tr}}$, and $a \leq t \leq b$.

Similar to the temporal authorization model, we associate a key $K^{x_{bl},y_{bl},a,x_{\mathrm{tr}},y_{\mathrm{tr}},b}$ with a spatial-temporal bounding box $(x_{bl}, y_{bl}, a, x_{\mathrm{tr}}, y_{\mathrm{tr}}, b)$. We use a broadcast protocol that is very similar to that used in the temporal authorization model in Section II. A broadcast includes $\langle x, y, t, E_{K^{x,y,t,x,y,t}}(P)\rangle$. Only an authorized subscriber can compute the encryption key $K^{x,y,t,x,y,t}$ and thus decrypt the broadcast packet P. We construct the keys such that:

- Given $K^{x_{bl},y_{bl},a,x_{\mathrm{tr}},y_{\mathrm{tr}},b}$, a user u can efficiently derive $K^{x,y,t,x,y,t}$ for all $x_{bl} \leq x \leq x_{\mathrm{tr}}$ and $y_{bl} \leq y \leq y_{\mathrm{tr}}$ and $a \leq t \leq b$.
- Given $K^{x_{bl},y_{bl},a,x_{\mathrm{tr}},y_{\mathrm{tr}},b}$, it is computationally infeasible for a user u to guess $K^{x,y,t,x,y,t}$ if $x < x_{bl}$, $x > x_{\mathrm{tr}}$, $y < y_{bl}$, $y > y_{\mathrm{tr}}$, $t < a$, or $t > b$.

B. Key Management Algorithm

Let us suppose that $X^1, X^2, \ldots X^d$ denote the d orthogonal domains. Without loss of generality, we assume that the minimum and maximum values from a domain i are 0 and $X^i_{\max}$, respectively. We construct a key tree starting from the root element $(0, 0, \ldots, 0, X^1_{\max}, X^2_{\max}, \ldots X^d_{\max})$. We divide each element $(X^1_a, X^2_a, \ldots X^d_a, X^1_b, X^2_b, \ldots X^d_b)$ into 2^d elements as follows. The bottom left corner of these 2^d bounding boxes can be compactly represented as a Cartesian product as: $\{X^1_a, \frac{X^1_a + X^1_b}{2}\} \times \{X^2_a, \frac{X^2_a + X^2_b}{2}\} \times \cdots \times \{X^d_a, \frac{X^d_a + X^d_b}{2}\}$. Each bounding box is for size $(\frac{X^1_b - X^1_a}{2}, \frac{X^2_b - X^2_a}{2}, \ldots, \frac{X^d_b - X^d_a}{2})$. Given the lower left corner and the size of each bounding box, one can easily determine the top right corner. For each of these 2^d bounding boxes, we derive keys as follows: $K^{X'^1_a, X'^2_a, \ldots, X'^d_a, X'^1_b, X'^2_b, \ldots, X'^d_b} =$

TABLE IX
QUANTIFICATIONS: COMMUNICATION COST

	Join (KDC)	Join (user)	Leave (KDC/user)
∀ (max)	$(2\log_2 \frac{T_{max}}{\delta t} - 2)K$	$(2\log_2 \frac{T_{max}}{\delta t} - 2)K$	-
∀ (avg)	$(\log_2 \frac{b-a}{\delta t})K$	$(\log_2 \frac{b-a}{\delta t})K$	-
∃ (max)	$(4\log_2 \frac{T_{max}}{\delta t} - 3)K$	$(4\log_2 \frac{T_{max}}{\delta t} - 3)K$	-
∃ (avg)	$(2 * \log_2 \frac{b-a}{\delta t} + 1)K$	$(2 * \log_2 \frac{b-a}{\delta t} + 1)K$	-

TABLE X
QUANTIFICATIONS: COMPUTATION COST

	Join (KDC)	Join (user)	Leave (KDC/user)	Key Derivation
∀ (max)	$(4\log_2 \frac{T_{max}}{\delta t} - 2)H$	-	-	$(4\log_2 \frac{b-a}{\delta t} - 2)H$
∀ (avg)	$(\log_2 \frac{T_{max}}{\delta t} + \log_2 \frac{b-a}{\delta t} - 1)H$	-	-	$(\log_2 \frac{b-a}{\delta t} + \log_2 \frac{end-beg}{\delta t} - 1)H$
∃ (max)	$(6\log_2 \frac{T_{max}}{\delta t} - 3)H$	-	-	$(\log_2 \frac{b-a}{\delta t} + 1)H$
∃ (avg)	$(\log_2 \frac{T_{max}}{\delta t} + 2 * \log_2 \frac{b-a}{\delta t})H$	-	-	$(\log_2(b-a) - \log_2 \lvert(a,b) \cap (beg,end)\rvert + 1)H$

TABLE XI
STORAGE COST

	KDC	User
TAC	$(X_{max} \log\log X_{max})^d$	$2^d * K$
STauth(max)	K	$2^d(2 * \frac{\sum_{i=1}^{d} \log_2 X^i_{max}}{d} - 1) * K$
STauth(avg)	K	$2^{d-1}(\frac{\sum_{i=1}^{d} \log_2 x^i}{d}) * K$

TABLE XII
COMMUNICATION COST

	Join (KDC/User)	Msg (user)
TAC	$2^d * K$	$2^{d+1} * K$
STauth(max)	$2^d(2 * \frac{\sum_{i=1}^{d} \log_2 X^i_{max}}{d} - 1) * K$	-
STauth(avg)	$2^{d-1}(\frac{\sum_{i=1}^{d} \log_2 x^i}{d}) * K$	-

TABLE XIII
COMPUTATION COST

	Join (KDC)	Join (User)	Terminate (KDC/user)	Msg (User)
TAC	-	-	-	$2^d * H + D$
STauth (max)	$2^d(2 * \frac{\sum_{i=1}^{d} \log_2 X^i_{max}}{d} - 1) * H$	-	-	$2^d(2 * \frac{\sum_{i=1}^{d} \log_2 x^i}{d} - 1) * H + D$
STauth (avg)	$2^{d-1}(\frac{\sum_{i=1}^{d} \log_2 X^i_{max}}{d} + \frac{\sum_{i=1}^{d} \log x^i}{d} - 1) * H$	-	-	$2^d(2 * \frac{\sum_{i=1}^{d} \log_2 x^i_{cache}}{d} - 1) * H + D$

$H(K^{X^1_a,X^2_a,\ldots,X^d_a,X^1_b,X^2_b,\ldots,X^d_b}, \xi_1\xi_2\cdots\xi_d)$, where $\xi_i = 0$ if $X'^i_a = X^i_a$ and $\xi_i = 1$, otherwise.

Tables XI–XIII respectively show the storage, communication, and computation cost incurred by our approach. Note that the costs tend to grow exponentially in the number of dimensions d. For typical spatial-temporal-based LBS applications, $d = 3$, and thus the cost of our key management algorithms would be acceptably small. Note that x^i denotes the extent of an authorization on the ith domain, and $(x^1_{\text{cache}}, x^2_{\text{cache}}, \ldots, x^d_{\text{cache}})$ denotes the size of the smallest cached bounding box that includes the d-dimensional coordinate in the broadcast message.

C. Comparison With Group Key Management Approaches

In this section, we analytically compare the communication cost incurred by the key management server using our approach and the group key management approach. Let us suppose that there are N active subscribers in the system. When a new user u joins the system, the key management server needs to update the group keys of all those users whose bounding box overlaps with that of user u. Let us suppose that $(x^1, x^2, \ldots, x^d)$ denote the average size of a subscription range along the d-dimensions. The subscription range along the ith dimension is assumed to be chosen uniformly and randomly from $(0, X^i_{\max})$. Hence, the probability that a subscription range of the new user u overlaps with an active user u' in the ith dimension is $\frac{2x^i}{X^i_{\max}}$ (if, $x^i < \frac{X^i_{\max}}{2}$). Note that if $x^i \geq \frac{X^i_{\max}}{2}$, then the probability of overlap is one. The bounding boxes for a user u and a user u' overlap if their subscriptions overlap on all the d-dimensions. Hence, the probability that the bounding box of a new user u overlaps with some active user u' is given by

$$\Pr_{\text{overlap}} = \frac{\prod_{i=1}^{d}(2x^i)}{\prod_{i=1}^{d}(X^i_{\max})} = 2^d \prod_{i=1}^{d} \frac{x^i}{X^i_{\max}}. \quad (5)$$

Therefore, the key update cost is $N * \Pr_{\text{overlap}}$. For every user u' whose subscription range overlaps with user u, the key server has to break up the bounding box into an average of 2^d subboxes. Fig. 3 illustrates the creation of new subboxes as new users join the system for a $(d = 2)$-dimensional domain. The size of the

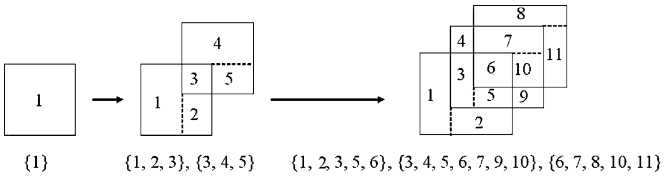

Fig. 3. User join: Group key management.

TABLE XIV
$d = 3, \frac{x}{X_{\max}} = 0.1$

N	x
10	1.12
10^2	3.03
10^3	2^{16}
10^4	2^{160}
10^5	2^{1600}

TABLE XV
$d = 3, N = 10^3$

$\frac{x}{X_{max}}$	x
0.01	1
0.05	4
0.1	2^{16}
0.15	2^{54}
0.20	2^{128}

average key update message for every overlapping user u' is 2^d keys. Therefore, the total cost of a new user join using the group key management is

$$\text{cost}_{\text{gkm}} = 2^d * N * 2^d \prod_{i=1}^{d} \frac{x^i}{X_{\max}^i}. \tag{6}$$

The cost of a new user join in our key management protocol is $\text{cost}_{\text{our}} = 2^{d-1} * \frac{\sum_{i=1}^{d} \log x^i}{d}$. The ratio of the costs is

$$\text{cost}_{\text{gkm}} : \text{cost}_{\text{our}} = \frac{2^{d+1} * N * d}{\sum_{i=1}^{d} \log x^i} * \prod_{i=1}^{d} \frac{x^i}{X_{\max}^i}. \tag{7}$$

Let us suppose that the subscription range along each dimension $x^i = x$, and the maximum subscription range along each dimension $X_{\max}^i = X_{\max}$. Then, the ratio becomes $\frac{2^{d+1} * N}{\log x} * (\frac{x}{X_{\max}})^d$. Now, setting $N = 10^4, d = 3$, and $\frac{x}{X_{\max}} = 0.1$, we observe that $\text{cost}_{\text{gkm}} : \text{cost}_{\text{our}}$ is smaller than 1 only if $x \geq 2^{160}$. Tables XIV and XV show the maximum value of x for $(d = 3)$-dimensional domain such that $\text{cost}_{\text{our}} \leq \text{cost}_{\text{gkm}}$ for different values of N and $\frac{x}{X_{\max}}$.

V. Partial-Order Trees

A. Overview

So far, we have studied applications of our key management to multidimensional domains wherein each domain has a well-defined total order. One can easily extend our algorithm to operate on domains that only have a partial order defined on them. We motivate an application of our algorithm using spatio-quality access control on services like Google Earth. Informally, a spatio-quality authorization is specified by a five tuple: $(x_{bl}, y_{bl}, x_{\text{tr}}, y_{\text{tr}}, q)$, where $(x_{bl}, y_{bl}, x_{\text{tr}}, y_{\text{tr}})$ denotes the spatial bounding box, and q denotes a quality of the image. Typically, one can build a partial-order tree on the quality q by moderating the resolution, smoothness of the image. A satellite image of the Earth broadcast by a public service should be intelligible to a user only if the coordinates of the broadcast image is within $(x_{bl}, y_{bl}, x_{\text{tr}}, y_{\text{tr}})$ and the image quality $q' \leq q$, where $\leq$ is overloaded to operate on the partial order defined on the quality domain.

Similar to the multidimensional authorization model, we associate a key $K^{x_{bl}, y_{bl}, x_{\text{tr}}, y_{\text{tr}}, q}$ with a spatial-quality bounding box $(x_{bl}, y_{bl}, x_{\text{tr}}, y_{\text{tr}}, q)$. We use a broadcast protocol similar to Section II. A broadcast packet includes $\langle x, y, q, E_{K^{x,y,x,y,q}}(P)\rangle$. Only an authorized subscriber can compute the decryption key $K^{x,y,x,y,q}$ and thus decrypt the broadcast packet P. We construct the keys such that:

- Given $K^{x_{bl}, y_{bl}, x_{\text{tr}}, y_{\text{tr}}, q}$, a user u can efficiently derive $K^{x,y,x,y,q'}$ for all $x_{bl} \leq x \leq x_{\text{tr}}$, $y_{bl} \leq y \leq y_{\text{tr}}$, and $q' \leq q$.
- Given $K^{x_{bl}, y_{bl}, x_{\text{tr}}, y_{\text{tr}}, q}$, it should be computationally infeasible for a user u to guess $K^{x,y,x,y,q'}$ if $x < x_{bl}$, $x > x_{\text{tr}}$, $y < y_{bl}$, $y > y_{\text{tr}}$, or $q > q'$.

B. Key Management Algorithm

The key idea of our approach is to map the partial-order tree into a totally ordered numeric range $(0, 2^s - 1)$, where s is a sufficiently large integer. Let x_0 denote the root element in the partial order. If the partial order has more than one root element, we follow the same procedure for every such maximal root element. We associate a totally ordered numeric range with every element in the partial order. First, we associate the range $(0, 2^s - 1)$ with the root element x_0. We define a minimal submissive set for every element x in the partial-order domain PO as $\text{minSub}(x)$

$$\text{sub}(x) = \{y : x > y\}$$
$$\text{minSub}(x) = \{y : y \in \text{sub}(x) \wedge (\nexists y' \in \text{sub}(x), y' > y)\}.$$

We partition the range associated with x for each element $y \in \text{minSub}(x)$. Let (x_a, x_b) denote the range associated with the element x. We partition this range into $2^{\lceil \log_2 |\text{minSub}(x)| \rceil}$ equally sized subranges and associate a distinct subrange with every element $y \in \text{minSub}(x)$. We repeat this process recursively starting from the root element x_0 and its associated range $(0, 2^s - 1)$. The range assignment maintains the property that $x > y$ if and only if $x_a \leq y_a \leq y_b \leq x_b$. Fig. 4 illustrates range assignment for a small partial-order domain $q \in \{A, B, \ldots, H\}$ such that $A > \{B, C\}, B > \{D, E, F\}$ and $C > \{H\}$.

We associate a key K^{x_a, x_b} with the element x. We derive this key from the root key, namely $K^{0, 2^s - 1}$, using the same recursive formulas shown in (1). This key derivation ensures that K^{y_a, y_b} can be efficiently derived from K^{x_a, x_b} if and only if $x_a \leq y_a \leq y_b \leq x_b$. Combining this with our assignment of ranges to each element in the partial-order tree, one can show that K^{y_a, y_b} can be efficiently derived from K^{x_a, x_b} if and only if $y \leq x$ in the partial-order domain. Fig. 4 shows the assignment of authorization keys in a partial-order domain. Table XVI shows the computation cost of our approach, where $d(x)$ denotes the depth of $x \in \text{PO}$ on the partial-order tree; note that

TABLE XVI
COMPUTATION COST

	Join (KDC)	Join (User)	Leave (KDC/User)	Msg (User)
(max)	$Max_{x \in PO} d(x) * H$	-	-	$Max_{x \in PO} d(x) * H + D$
(avg)	$\sum_{x \in PO} (d(x) * f(x)) * H$	-	-	$\sum_{y \leq x} ((d(y) - d(x)) * f(y)) * H + D$

TABLE XVII
$N = 10^2, |\mathrm{PO}| = 100$

Distribution	Parameter	$cost_{gkm} : cost_{our}$
Geometric	$\frac{1}{p}$ = 0.01$\|PO\|$	199
Geometric	$\frac{1}{p}$ = 0.1$\|PO\|$	30.4
Geometric	$\frac{1}{p}$ = 0.5$\|PO\|$	6.7
Geometric	$\frac{1}{p}$ = $\|PO\|$	2.1
Gaussian	μ = 0.5$\|PO\|$, σ = 0.01$\|PO\|$	296
Gaussian	μ = 0.5$\|PO\|$, σ = 0.1$\|PO\|$	162
Gaussian	μ = 0.5$\|PO\|$, σ = 0.5$\|PO\|$	50
Zipf	γ = 0.01	8.4
Zipf	γ = 0.1	12.8
Zipf	γ = 0.5	80

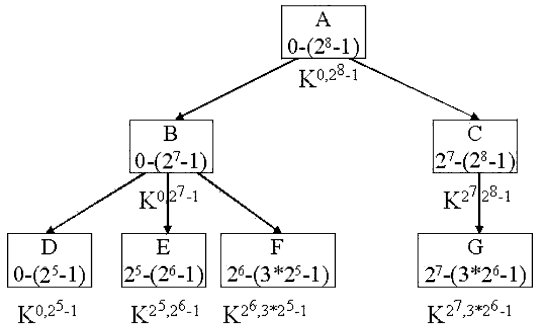

Fig. 4. Partial-order trees.

the computation and storage cost at both the KDC and user is K.

C. Comparison With Group Key Management Approaches

A cost analysis for our key management algorithm on partial domains is similar to that for a totally ordered domain. We compute the probability that the subscription of a new user u overlaps with some active user u' as $\mathrm{Pr}_{\mathrm{overlap}}$ in

$$\mathrm{Pr}_{\mathrm{overlap}} = \sum_{x \in \mathrm{PO}} f(x) * \left(\sum_{y \leq x} f(y) + \sum_{y > x} f(y) \right). \quad (8)$$

We use $f(x)(x \in \mathrm{PO})$ as the probability distribution of user subscriptions over the partially ordered domain PO. Then, following the same lines of argument as in Section IV (using $d = 1$), one can show that cost ratio of our approach $(\mathrm{cost}_{\mathrm{our}})$ to the group key management approach $(\mathrm{cost}_{\mathrm{gkm}})$ is

$$\mathrm{cost}_{\mathrm{gkm}} : \mathrm{cost}_{our} = \frac{2 * N * Pr_{\mathrm{overlap}}}{\overline{s}} \quad (9)$$

where $\overline{s}$ is the average height of the partial-order tree. The cost ratio as shown in (9) attains a minimum value when $\mathrm{Pr}_{\mathrm{overlap}}$ is minimum. Evidently, this is achieved when, for any $x, y \in \mathrm{PO}$ such that $x \neq y$, neither $x < y$ nor $y < x$. Hence, $\mathrm{Pr}^{\min}_{\mathrm{overlap}}$ is given by $\mathrm{Pr}^{\min}_{\mathrm{overlap}} = \sum_{x \in \mathrm{PO}} f(x)^2$. Given that $\sum_{x \in \mathrm{PO}} f(x) = 1$, one can show that $\mathrm{Pr}^{\min}_{\mathrm{overlap}}$ achieves an absolute minima when $f(x)$ is uniformly and randomly distributed; that is, $f(x) = \frac{1}{|\mathrm{PO}|}$ for all $x \in \mathrm{PO}$, where $|\mathrm{PO}|$ denotes the size of the partial-order domain PO. The absolute minima $\mathrm{Pr}^{\mathrm{minima}}_{\mathrm{overlap}}$ is given by $\mathrm{Pr}^{\mathrm{minima}}_{\mathrm{overlap}} = \frac{1}{|\mathrm{PO}|}$.

However, assuming that no two $x, y \in \mathrm{PO}$ are related by the operator $<$, using a uniform and random distribution of $f(x)$ may not be realistic. We relax the first constraint and assume that $f(x)$ is uniform and random and that the partial-order tree is a r-ary tree of height s. One can show that $\sum_{y \leq x} f(y) = \sum_{i=d(x)}^{\log_r |\mathrm{PO}|} r^i = \frac{r^{\log_r |\mathrm{PO}| - d(x) + 1} - 1}{r-1}$ and $\sum_{y > x} f(y) = d(x)$, where $d(x)$ denotes the depth of x in the partial order $(d(\mathrm{root}) = 0)$. Hence, the probability of overlap in a r-ary tree is

$$\begin{aligned} \mathrm{Pr}^{r-\mathrm{ary\ tree}}_{\mathrm{overlap}} &= \sum_{i=0}^{\log_r |\mathrm{PO}|} \frac{r^i}{|\mathrm{PO}|} * \frac{\frac{|\mathrm{PO}| * r^{-i+1} - 1}{r-1} + i}{|\mathrm{PO}|} \\ &= \frac{2 \log_r |\mathrm{PO}| - 1}{|\mathrm{PO}|} + 2 * \frac{\log_r |\mathrm{PO}| + 1}{|\mathrm{PO}|^2}. \end{aligned} \quad (10)$$

Observe that this probability is $2 \log_r |\mathrm{PO}| - 1$ times larger than the absolute minima. Note that as r increases, the probability of overlap decreases. However, the height of an r-ary tree is $\overline{s} = \log_r |\mathrm{PO}|$. Plugging this into (9), the cost ratio between the group key management protocols and our approach is: $\mathrm{cost}_{\mathrm{gkm}} : \mathrm{cost}_{\mathrm{our}} \approx \frac{4 * N}{|\mathrm{PO}|}$. Observe that the cost ratio is independent of the parameter r.

Now, we relax the second constraint and assume that no $x, y \in \mathrm{PO}$ are related by the operator $<$ while using nonuniform distributions (like truncated Geometric, discrete approximation to Gaussian and Zipf) for the function f. Table XVII summarizes our results for different parameters of these distributions for $N = 100$ and $|\mathrm{PO}| = 100$. Observe that as the standard deviation increases, the probability of overlap between two subscriptions decreases, thereby reducing the cost of the group key management algorithms. One the other hand, our approach incurs a small and a constant (nearly) cost that is completely agnostic to the distribution of user subscriptions.

VI. EXPERIMENTAL EVALUATION

We have implemented our key management algorithms on the Siena publish-subscribe network [15]. Siena is a wide-area publish-subscribe network that allows events to be disseminated from an LBS server (publisher) to a geographically scattered group of subscribers. We used GT-ITM [34] topology generator to generate an Internet topology consisting of 63 nodes. The round-rip times on these links varied from 24–184 ms with mean 74 ms and standard deviation 50 ms. We constructed a

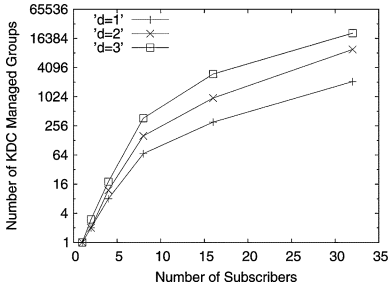

Fig. 5. Scalability issue with group key management protocols.

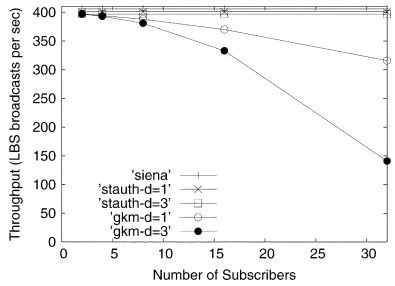

Fig. 6. Throughput versus number of subscribers.

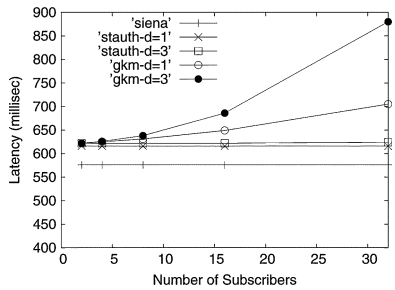

Fig. 7. Latency versus number of subscribers.

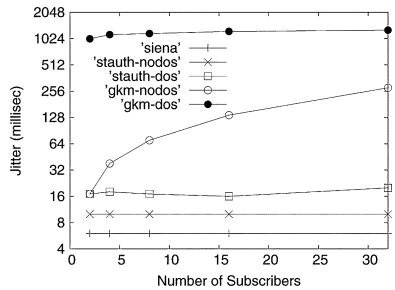

Fig. 8. Resilience to DoS attacks.

complete binary tree topology using 63 nodes. The tree's root node acts as the LBS server, 32 leaf nodes act as subscribers, and 30 nodes operate as routing nodes. We ran our implementation of `STauth` on eight 8-processor servers (64 CPUs) (550-MHz Intel Pentium III Xeon processors running RedHat Linux 9.0) connected via a high-speed LAN. We simulated the wide-area network delays obtained from the GT-ITM topology generator.

All experimental results presented in this section were averaged over five independent runs; each run represents an hour-ong experiment on our publish-subscribe network that measures various performance metrics such as throughput and response time. We simulated a spatial-temporal space of volume $1024 \times 1024 \times 1024$. The size of a subscription range (along each dimension) was chosen using a Gaussian distribution with mean 256 and a standard deviation 64. The subscription boxes (left bottom corner) for the spatial coordinates were chosen using a 2-dimensional Gaussian distribution centered at coordinate (512, 512), while that for the temporal coordinate was chosen uniformly and randomly over (0, 1024). Each LBS broadcast message was assumed to be of size 1 kB.

In this section, we show two experimental results. First, we compare our proposals with traditional group key management approaches. We demonstrate the scalability problems in group key management protocols by measuring the number of groups that need to be managed by the KDC; we measure the overhead of our algorithms over the insecure LBS system in terms of its throughput and latency; and we demonstrate the low jitter and purported future keys based DoS attack resilience properties of our protocols in comparison with the group key management protocols. Second, we compare our approach with recently proposed key management algorithms [7]–[9] for temporal and geospatial access control. Third, we describe an implementation of our spatial-quality key management algorithms using the Google Maps API. We show that our approach incurs minimally on page load time while enforcing spatial-quality access control on maps.

A. Group Key Management Protocols

Scalability. Fig. 5 demonstrates the lack of scalability in traditional group key management protocols. The figure shows the number of groups that need to be managed by the KDC versus the number of subscribers N for different values of dimensionality d. Even for 32 subscribers, the number of managed groups may be of the order of 10^4 with $d = 3$. Our analysis indicates that even for a modest set of 1000 subscribers, the number of managed groups could be about 2^{112}.

Throughput and Latency. Figs. 6 and 7 show the throughput and latency of LBS broadcasts, respectively. We observe that the throughput loss due to our key management algorithm is very small when compared to the insecure Siena network. The increase in latency due to our key management algorithm can be attributed almost entirely to the encryption and decryption costs; the key management costs account for less than 12% of the overhead. Traditional group key management protocols on the other hand incur a significant drop in throughput (62.5% for $N = 32$) and increase in latency as the number of subscribers increases (52% for $N = 32$). Our simulation results indicate that for $N = 1000$ subscribers, the throughput drop is about 99.96% and the increase in latency is about 140 times.

DoS Attack. Fig. 8 shows the jitter (standard deviation in interpacket arrival times) in LBS broadcasts. The jitter added by our key management protocol, even when under a DoS attack (purported future-key-based DoS attack), is only a few tens of milliseconds, which is less than 3% of the mean latency. On the other hand, the jitter incurred by traditional group key management protocols, even in the absence of DoS attacks, is about 22%, and under a DoS attack is about 200%. This clearly demonstrates the vulnerability of traditional group key management protocols to the purported future-key-based DoS attack.

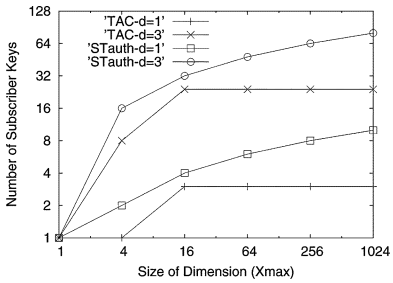

Fig. 9. Number of subscriber keys.

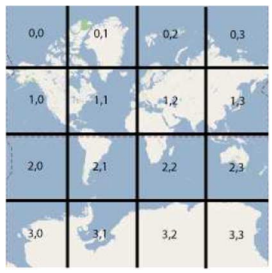

Fig. 10. Zoom level 2: 16 tiles.

Fig. 11. Zoom level 12: Access permitted.

B. Temporal and Geospatial Access Control

In this section, we compare our key management algorithms with other hierarchical key derivation algorithms (TAC [7]–[9]).

Number of Keys. Fig. 9 shows the number of keys maintained by a subscriber (incurred by both STauth and TAC). We observe that the number of subscriber keys incurred by the TAC is a constant, while that in the STauth approach grows logarithmically with the size of the dimension $X_{\max}$. Hence, TAC requires a subscriber to maintain fewer keys. Fig. 13 shows the size of public storage incurred by TAC; note that STauth requires no public storage. TAC requires public storage whose size is at least proportional $X_{\max}^d$ (where d is the number of dimensions). Fig. 13 shows that the size of public storage can grow prohibitively large for large dimensions $(X_{\max})$ and the number of dimensions (d).

Key Derivation Cost. Figs. 14 and 15 show the computation and communication cost incurred during key derivation. Recall that the key derivation cost is incurred on the receipt of each broadcast packet. We observe that the STauth approach incurs marginally higher computation cost (in microseconds). Furthermore, one can use the key-caching-based approach described in Section II to reduce the key derivation computation cost to a small constant in the STauth approach. On the other hand, TAC incurs communication cost during key derivation; though the per-subscriber communication is small (few 100 bytes), the aggregate load on the public storage grows linearly with the number of subscriber in the network. This can additionally increase application-level latency and jitter and may render the network vulnerable to DoS attacks on public storage.

C. Spatial-Quality Access Control

In this section, we describe an implementation of our algorithms for spatial-quality key management (see Section V) on Google Maps [3]. Recall that a spatial-quality authorization is specified by a five tuple: $(x_{bl}, y_{bl}, x_{\mathrm{tr}}, y_{\mathrm{tr}}, q)$, where $(x_{bl}, y_{bl}, x_{\mathrm{tr}}, y_{\mathrm{tr}})$ denotes the spatial bounding box, and q denotes quality (of the map in this scenario). We implemented STauth using JavaScripts (AJAX model) that export three interfaces: `boolean boundingBox (coordinates, quality, authBox)` checks if `(coordinates, quality)` of a tile file belongs to the client's spatial-quality authorization box `authBox`; `key deriveKey (coordinates, quality)` derives the decryption key for a given coordinate and quality tuple; and `boolean decryptImage (map, key)` decrypts the map image using `key`.

Before we describe our implementation, we provide a brief overview of Google Maps. There are three coordinates in Google Maps: tile, pixel, and zoom level. Google Maps divides the entire Earth into multiple square tiles. Each tile consists of 256×256 pixels irrespective of the zoom level. At zoom level n, the Earth is divided into 4^n tiles $(1 \leq n \leq 19)$. When transitioning from zoom level n to $n+1$, each tile is divided into four quadrants, thereby, doubling the pixel space in both the x- and y-axis. Fig. 10 shows that at zoom level 2, the Earth is divided into $16(= 4^2)$ tiles. We note that the way Google Maps divides Earth into tiles is exactly identical to our approach of defining and partitioning a 2-dimensional bounding box (X_a, Y_a, X_b, Y_b). We treat the zoom level as a totally ordered quality dimension; the higher the zoom level, the better the quality.

Fig. 16 shows a code snippet of a JavaScript-based implementation of our access control algorithm using the Google Maps API. The Web server (Apache HTTPD [1]) serves tiles as image files; tiles are indexed by their center (latitude, longitude) and the zoom level. The server applies our key management algorithms to derive the encryption key for each tile and encrypts the tile (image file) with the corresponding key. In response to a client's (Web browser: FireFox or Microsoft IE) request, the server returns an encrypted tile file. The client checks if the tile belongs to its authorized bounding box (`authBox`). If so, the client derives the decryption key, decrypts the tile file, and renders the image (see Fig. 11); otherwise, the client throws an alert (see Fig. 12) indicating that the user is not authorized to view the tile (at the requested zoom level).

Our initial experiments indicate the percentile overhead added by our key management algorithms to the page load time is about 0.72% (indicating that our key derivation cost is very small). We also used a client side JavaScript to draw random tiles and measured the throughput [number of Web pages per second (WPP)]. We measured the drop in throughput at the client as 0.4% and 0.44% using Mozilla FireFox and Microsoft IE, respectively.

Fig. 12. Zoom level 13: Not authorized.

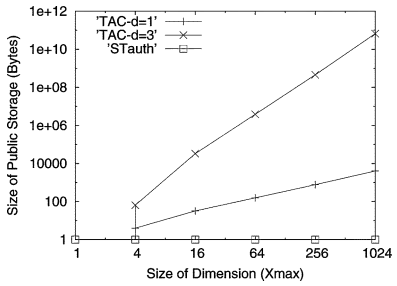

Fig. 13. TAC: Size of public storage.

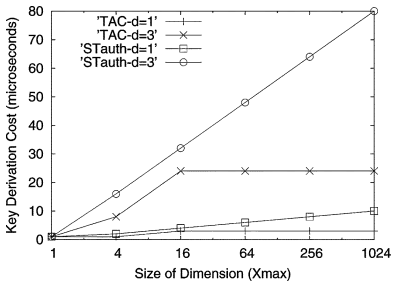

Fig. 14. Key derivation computation cost.

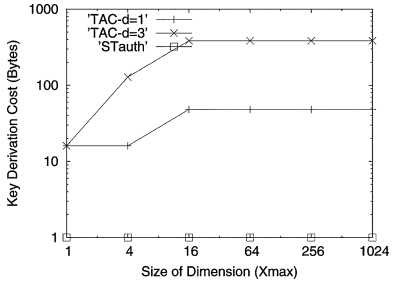

Fig. 15. Key derivation communication cost.

We note that the size of the spatial-quality dimension in Google Maps is $2^{19} \times 2^{19} \times 19$. While the total number keys managed by the system is $19 * 2^{38} = 5.22 * 10^{12}$, STauth incurs low overhead primarily because of its efficient key derivation algorithm. The TAC approach incurs slightly lower key derivation computation cost than the STauth approach; however, the size of public storage using the TAC approach is $2^{38} * 19 * 16$ bytes $= 76$ TB. Hence, maintaining integrity of public storage data and serving the data in real-time pose severe challenges for the TAC approach.

```
function load() {
   GEvent.addListener(map, "click", function() {
   //Left Click = Zoom In
   var center = map.getCenter(); var zoom = map.getZoom() + 1;
   if(boundingBox(center, zoom, authBox)) {
     var dKey = deriveKey(center, zoom);
     map.setCenter(center, zoom); //gets encrypted file
     if(!decryptImage(map, dKey))
       alert("Integrity check on map failed");
   } else {
     alert("No auth key found for zoom level: " + ···)
   } } }

   var tileLayerOverlay = new GTileLayerOverlay(
     new GTileLayer(null, null, null, {
     //Get encrypted tile image: center = (X, Y) and zoom = Z
     tileUrlTemplate: 'http://nc12.watson.ibm.com/cryptImg-{Z}-{X}-{Y}.png',
     isPng:true, opacity:1.0});
   map.addOverlay(tileLayerOverlay);
}

<body onload="load()" onunload="GUnload()">
   <div id="map" style="width: 256px; height: 256px"></div>
</body>
```

Fig. 16. Spatial-quality access control using Google Maps API: JavaScript code snippets.

VII. Related Work

Broadcast encryption [22] is the problem of sending an encrypted message to a large user base (size N) such that the message can only be decrypted by a dynamically changing privileged subset (size $N - r$). However, such schemes are designed to operate in scenarios where $r \ll N$; for example, optimal LSD [19] broadcast encryption scheme requires message headers of size $O(r \log \log N)$. In the context of LBS, r can be $O(N)$, making it nontrivial to use traditional broadcast encryption schemes. More recently, Boneh *et al.* [11] have proposed efficient schemes for subsets of arbitrary sizes. However, their scheme still requires a message header of size $O(\sqrt{N})$ and incurs the overhead of expensive pairing operations. In the context of LBS, the broadcast messages are typically very small; in energy-constrained wireless environments, it is important to restrict message headers to $O(1)$ size.

Group key management addresses the problem of sending an encrypted message to a large and dynamic user base such that the message can only be decrypted by the members of the user base. In contrast to broadcast encryption, the parameter N dynamically changes in a group key management system; however, there is no concept of a privileged subset of users in the group. Significant amount of work has been done in the field of group key management using the concept of a logical key hierarchy [20]. Several papers [6], [13], [14], [24], [26], [29], [30], [32] have developed interesting optimization techniques to enhance the performance and scalability of group key management protocols on multicast networks. Some extensions to operate on unreliable multicast channels are proposed in [26] and [33]. A detailed survey along with comparisons among various group key management protocols is described in [27]. Group key management protocols use message headers of constant size (unlike $O(\sqrt{N})$ in broadcast encryption), making them more suitable for our target application. However, the cost of key management (as demonstrated in Section VI) in the context of LBS is unacceptably high.

Recently, several papers [7], [10], [16], [28] have proposed to exploit the hierarchical structure of an authorization model to develop more efficient key management schemes. Similar to [7]

(which we compare to in this paper), the other schemes require public storage that is at least linear in the size of the authorization space. STauth also exploits the hierarchical structure of the authorization model; it builds on the MARKS protocol [12] and requires no public storage.

VIII. Conclusion

In this paper, we have presented STauth, a scalable key management algorithm for enforcing spatial-temporal access control on public broadcast services. Unlike traditional group key management approaches, we exploit the spatial-temporal authorization model to construct authorization keys using efficient and secure hierarchical key graphs. We have shown that our approach solves several drawbacks in traditional group key management approaches, including poor scalability, vulnerability to packet losses, failures in the presence of packet losses, vulnerability to certain DoS attacks, and susceptibility to jitters and delays. We have described a prototype implementation and experimental evaluation that demonstrate our performance and scalability benefits while preserving the security guarantees.

References

[1] Apache HTTPD server. [Online]. Available: http://www.apache.org/
[2] Garmin. [Online]. Available: http://www.garmin.com ,
[3] Google Maps API. [Online]. Available: http://code.google.com/apis/maps/
[4] Loc Aid. [Online]. Available: http://www.loc-aid.net
[5] Veripath Navigator. [Online]. Available: http://veripath.us
[6] K. Aguilera and R. Strom, "Efficient atomic broadcast using deterministic merge," in *Proc. 19th ACM PODC*, 2000, pp. 209–218.
[7] M. Atallah, K. Frikken, and M. Blanton, "Dynamic and efficient key management for access hierarchies," in *Proc. ACM CCS*, 2005, pp. 190–202.
[8] M. J. Atallah, M. Blanton, and K. B. Frikken, "Efficient techniques for realizing geo-spatial access control," in *Proc. Asia CCS*, 2007, pp. 82–92.
[9] M. J. Atallah, M. Blanton, and K. B. Frikken, "Incorporating temporal capabilities in existing key management schemes," in *Proc. ESORICS*, 2007, pp. 515–530.
[10] G. Ateniese, A. D. Santis, A. L. Ferrara, and B. Masucci, "Provably secure time bound hierarchical key assignment schemes," in *Proc. ACM CCS*, 2006, pp. 288–297.
[11] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Proc. Crypto*, 2005, pp. 258–275.
[12] B. Briscoe, "Marks: Zero side-effect multicast key management using arbitrarily revealed key sequences," in *Proc. 1st Workshop on Netw. Group Commun.*, 1999, pp. 301–320.
[13] R. Canetti, J. Garay, G. Itkis, and D. Micciancio, "Multicast security: A taxonomy and some efficient constructions," in *Proc. IEEE INFOCOM*, 1999, vol. 2, pp. 708–716.
[14] R. Canetti, T. Malkin, and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption," *EUROCRYPT, LNCS*, vol. 1599, pp. 459–474, 1999.
[15] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM TOCS*, vol. 19, no. 3, pp. 332–383, 2001.
[16] J. Crampton, K. Martin, and P. Wild, "On key assignment for hierarchical access control," in *Proc. Comput. Security Found. Workshop*, 2006, pp. 98–111.
[17] E. Eastlake, "US Secure Hash Algorithm I," 2001 [Online]. Available: http://www.ietf.org/rfc/rfc3174.txt,
[18] K. Fu, S. Kamara, and T. Kohno, "Key regression: Enabling efficient key distribution for secure distributed storage," in *Proc. NDSS*, 2005, online.
[19] D. Halevi and A. Shamir, "The LSD broadcast encryption scheme," in *Proc. Crypto*, 2002, pp. 145–161.
[20] H. Harney and E. Harder, "Logical key hierarchy protocol," [Online]. Available: http://www.rfc-archive.org/getrfc.php?rfc=2093
[21] H. Harney and C. Muckenhirn, "Group key management protocol (gKMP) architecture," [Online]. Available: http://www.rfc-archive.org/getrfc.php?rfc=2094
[22] J. Horowitz, "A survey of broadcast encryption," [Online]. Available: http://math.scu.edu/jhorwitz/pubs/broadcast.html
[23] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," [Online]. Available: http://www.faqs.org/rfcs/rfc2104.html
[24] D. A. McGrew and A. T. Sherman, "Key establishment in large dynamic groups using one-way function trees," TIS Labs at Network Associates, Inc., Glenwood, MD, Tech. Rep. No. 0755, May 1998.
[25] *NIST, AES: Advanced Encryption Standard*, [Online]. Available: http://csrc.nist.gov/CryptoToolkit/aes/
[26] A. Perrig, D. Song, and J. D. Tygar, "ELK: A new protocol for efficient large group key distribution," in *Proc. IEEE Symp. Security and Privacy*, 2001, pp. 247–262.
[27] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *J. ACM Comput. Surveys*, vol. 35, no. 3, pp. 309–329, 2003.
[28] A. D. Santis, A. L. Ferrara, and B. Masucci, "New constructions for provably secure time bound hierarchical key assignment schemes," in *Proc. SACMAT*, 2007, pp. 133–138.
[29] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The versakey framework: Versatile group key management," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 8, pp. 1614–1631, Aug. 9, 1999.
[30] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: Issues and architectures," RFC 2627, 1999.
[31] C. Wong, M. Gouda, and S. Lam, "Secure group communications using key graphs," in *Proc. ACM SIGCOMM*, 1998, pp. 68–79.
[32] C. K. Wong, M. G. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. Netw.*, vol. 8, no. 1, pp. 16–30, Feb. 2000.
[33] C. K. Wong and S. S. Lam, "Keystone: A group key management service," in *Proc. ICT*, 2000, pp. 19–25.
[34] E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM*, 1996, pp. 594–602.

Mudhakar Srivatsa received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Madras, India, and the Ph.D. degree in computer science from Georgia Institute of Technology, Atlanta.

He is a Research Scientist at IBM T. J. Watson Research Center, Yorktown Heights, NY, where he conducts research at the intersection of networking and security. His research interests primarily include security and reliability of large-scale networks, secure information flow, and risk management.

Arun Iyengar received the Ph.D. degree in computer science from Massachusetts Institute of Technology, Cambridge.

He does research and development in Web performance, distributed computing, and high availability at IBM T. J. Watson Research Center, Yorktown Heights, NY. He is Co-Editor-in-Chief of the *ACM Transactions on the Web*, Founding Chair of IFIP Working Group 6.4 on Internet Applications Engineering, and an IBM Master Inventor.

Jian Yin received the Ph.D. degree in computer science from the University of Texas at Austin.

He does research and development in Web performance, distributed computing, and high availability at IBM T. J. Watson Research Center, Yorktown Heights, NY.

Ling Liu received the Ph.D. degree in computer science from Tilburg University, Tilburg, The Netherlands.

She is an Associate Professor at the College of Computing, Georgia Institute of Technology, Atlanta. She directs the research programs in the Distributed Data Intensive Systems Lab (DiSL), examining research issues and technical challenges in building large-scale distributed computing systems that can grow without limits. Her research group has produced a number of software systems that are either open sources or directly accessible online, among which the most popular are WebCQ and XWRAPElite.

Dr. Liu is on the Editorial Board of several international journals, such as the IEEE Transactions on Knowledge and Data Engineering, the *International Journal of Very large Database Systems (VLDBJ)*, and the *International Journal of Web Services Research*.