A Scalable Method for Access Control in Location-Based Broadcast Services

Mudhakar Srivatsa[†]**, Arun Iyengar**[†]**, Jian Yin**[†] **and Ling Liu**[‡]
IBM T.J. Watson Research Center, Yorktown Heights, NY 10562[†]
College of Computing, Georgia Institute of Technology, Atlanta, GA 30332[‡]
{msrivats, aruni, jianyin}@us.ibm.com, lingliu@cc.gatech.edu

Abstract. 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 a LBS for a *time interval* (a, b) and a *spatial region* $(x_{bl},\ y_{bl},\ x_{tr},\ y_{tr})$ according to a 3-dimensional spatial-temporal authorization model. In this paper, we argue that current approaches to access control using group key management protocols are not scalable. Our proposal STauth minimizes the number of keys which needs to be distributed and is thus scalable to a much higher number of subscribers and the dimensionality of the authorization model. We analytically and experimentally demonstrate the performance and scalability benefits of our approach against other group key management protocols.

I. Introduction

The ubiquitous nature of the Internet has resulted in widespread growth and deployment of location based services (LBS) [1], [2], [3]. LBS (as the name indicates) provide information with spatial-temporal validity to potentially resource constrained wireless and mobile subscribers. Example services include: (i) list all Italian restaurants in midtown Atlanta, (ii) current traffic conditions at the junction of `peach tree parkway` and `peach tree circle`, (iii) 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_{tr}, y_{tr}) and a time interval (a, b); the subscription fee may be an arbitrary function, say $fee \propto (x_{tr} - x_{bl}) \times (y_{tr} - y_{bl}) \times (b - a)$. 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 $x_{bl} \leq x \leq x_{tr}$ and $y_{bl} \leq y \leq y_{tr}$ and $a \leq t \leq b$.

A common solution for controlling 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: (i) 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 [15], LKH [25], [14], ELK [21]. 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. (ii) Large number of groups (new problem specific to LBS-like services). Using a spatial-temporal authorization model, each unit of data broadcast by a 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 which 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 $\langle x,\ y,\ t\rangle$). We briefly summarize the drawbacks of existing key management protocols.

1) In the worst case, KDC manages $O(2^{dN})$ groups.
2) User join and leave requires the KDC has 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. Typically, these protocols use a counter to identify the group keys. Each time the group key is updated (say, due to a user join/leave), the counter is incremented. When an authorized user has a group key identified by counter c, and it receives a broadcast packet that is encrypted with a future group key identified by counter $c' > c$, the user buffers the packet until it

receives the key update messages corresponding to the future group key. The unauthorized users can launch a DoS attack on this buffer by flooding the broadcast channel with packets that are purportedly encrypted with future group keys.

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 multi-dimensional authorization model, we use a simple and yet powerful key management protocol using hierarchical key graphs [5], [8] 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.

II. One-Dimensional 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 $\langle t$, $E_{K^{t,t}}(P)$, $MAC_{K^{t,t}}(P)\rangle$.

$E_K(x)$ and $MAC_K(x)$ 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 $\langle t$, $E_{K^{t,t}}(P)$, $MAC_{K^{t,t}}(P)\rangle$. 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 [8], [25], [5] that satisfy the primitive described in Section II-A. We first introduce some notation and parameters used in our algorithm. Let $(0, T_{max})$ denote the time horizon of interest. Let δt seconds denote the smallest time granularity of interest. Let time equal to t denote the t^{th} time unit, where one unit time = δt seconds. 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 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:

$$\begin{aligned} K^{a,\frac{a+b}{2}}(S) &= H(K^{a,b}(S), 0) \\ K^{\frac{a+b}{2}+1,b}(S) &= H(K^{a,b}(S), 1) \end{aligned} \tag{1}$$

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 $K^{0,T_{max}}(S)$ = $H(MK, S)$. 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. Figure 1 illustrates the construction of our key tree assuming T_{max} = 31 time units. We derive $K^{0,31}(S)$ = $H(MK, S)$. 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.

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

TABLE I
NOTATION

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 II
MAXIMUM NUMBER OF KEYS AND COMPUTATION TIME

$b - a$	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

TABLE III
AVERAGE NUMBER OF KEYS AND COMPUTATION TIME WITH δt = 1 SECOND

	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`	$T \log \log T * K$	$3K$
`STauth (max)`	K	$(2 \log_2 \frac{T_{max}}{\delta t} - 2)K$
`STauth (avg)`	K	$\log_2 \frac{b-a}{\delta t} * K$

TABLE IV
STORAGE COST

	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 V
SECURITY PROPERTIES

	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 VI
COMMUNICATION 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 Figure 1). We provide temporal authorization for a time interval $(8, 19)$ by issuing two authorization keys $K^{8,15}(S)$ and $K^{16,19}(S)$. One can use proofs similar to that in [5] to show that our algorithm for constructing authorization keys indeed satisfies the required security property.

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 $r(\log_r(\frac{T_{max}}{\delta t}) - 1)$ subinterval. 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 an 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 ($< 1\mu s$ 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

	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)`	$(\log_2 \frac{T_{max}}{\delta t} + \log_2 \frac{b-a}{\delta t} - 1)H$	-	-	-	$-H \log_2(rate * \delta t) + D$

TABLE VII
COMPUTATION COST

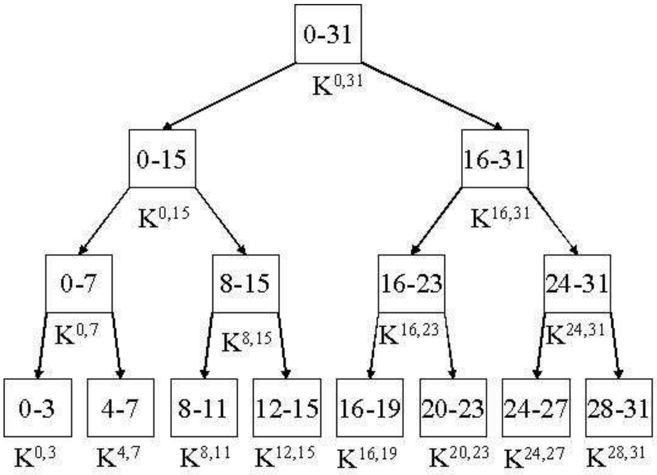

Fig. 1. Authorization Key Tree

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` [25] builds a logical key hierarchy on the set of authorized users to enhance the efficiency of the key update protocol. `ELK` [21] introduces the concepts of hints to enhance the efficiency of LKH protocol and improve its resilience to arbitrary packet loss of key update messages.

Atallah et. al. [5], [7], [6] (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 one-dimensional space (e.g.: time). The atomic primitive supported by their approach is to key derivation 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 to a user; 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 using a key caching based approach one can reduce the amortized key derivation cost to $O(1)$ in our approach. On the flip side, TAC incurs $O(1)$ communication cost for key derivation. We note that key derivation cost is incurred for every message received by the user. TAC requires at least $O(T * \log \log T)$ *public storage*. Using a fine grained access control (say, δt = one second), T 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, δt = $1 \mu s$). While public storage may be made available to all users (authorized or not) without compromising on 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 non-trivial 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 needs not 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` sends $O(\log_2 N)$ size message to all the users $O(N)$; and `ELK join` sends $O(\log_2 N)$ size message only to the new user while compromising backward secrecy for at most one *time interval*. Further, 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` needs 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 $K^{a,b}$ ($a \leq t \leq b$). This may require $\log_2 \frac{b-a}{\delta t}$ applications of H. Using standard cryptographic algorithms (say, HMAC-SHA [16], [12] for H and AES-CBC-128 [20] 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 denial of service attack (DoS) 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 MAC (message authentication code) $MAC_{K^{t,t}}(P)$ 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 $K^{a,b}$ ($a \leq t \leq b$), the user computes several intermediate keys $K^{a',b'}$ ($a \leq a' \leq t \leq b' \leq b$). 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 $a \leq a' \leq t \leq t' \leq b' \leq b$. Indeed, this would require only $\log_2 \frac{b'-a'}{\delta t}$ applications of H ($b' - a' \leq b - a$). One can show that if the mean inter-packet arrival time is $\frac{1}{rate}$ then, the mean per-packet key derivation cost drops to $-H \log_2(rate * \delta t)$ (assuming, $\delta t < \frac{1}{rate}$). An interesting observation is that the per-packet key derivation cost is independent of the length of the subscription interval $b - a$ (for reasonably large intervals (a, b)). Also, note that as $rate$ increases the per-packet key derivation cost decreases.

III. Multi-Dimensional Authorization

A. Overview

In this Section, we extend our key management algorithms to operate on multi-dimensional authorization models. We use location based services (LBS) as a motivating example. Location based services broadcast information with spatial-temporal validity, say, traffic information at the junction (x, y) at time t. An LBS service uses a spatial-temporal authorization model as follows: A user u subscribes for a spatial bounding box (x_{bl}, y_{bl}, x_{tr}, y_{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 $x_{bl} \leq x \leq x_{tr}$ and $y_{bl} \leq y \leq y_{tr}$ and $a \leq t \leq b$. Similar to the temporal authorization model, we associate a key $K^{x_{bl},y_{bl},a,x_{tr},y_{tr},b}$ with a spatial-temporal bounding box (x_{bl}, y_{bl}, a, x_{tr}, y_{tr}, b). We use a broadcast protocol that is very similar to that used in temporal authorization model in Section II. Each broadcast message includes $\langle\ x,\ y,\ t,\ E_{K^{x,y,t,x,y,t}}(P),\ MAC_{K^{x,y,t,x,y,t}}(P)\ \rangle$. 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 $K^{x_{bl},y_{bl},a,x_{tr},y_{tr},b}$ a user u can efficiently derive $K^{x,y,t,x,y,t}$ for all $x_{bl} \leq x \leq x_{tr}$ and $y_{bl} \leq y \leq y_{tr}$ and $a \leq t \leq b$.
- Given $K^{x_{bl},y_{bl},a,x_{tr},y_{tr},b}$ it is computationally infeasible for a user u to guess $K^{x,y,t,x,y,t}$ if $x < x_{bl}$ or $x > x_{tr}$ or $y < y_{bl}$ or $y > y_{tr}$ or $t < a$ or $t > b$.

B. Key Management Algorithm

Let us suppose that X^1, X^2, $\cdots$ X^d denote the d orthogonal dimensions. Without loss of generality we assume that the minimum and maximum values for a dimension i is 0 and X^i_{max} respectively. We construct a key tree starting from the root element (0, 0, $\cdots$, 0, X^1_{max}, X^2_{max}, $\cdots$ X^d_{max}). We divide each element (X^1_a, X^2_a, $\cdots$ X^d_a, X^1_b, X^2_b, $\cdots$ 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 $(\frac{X^1_b-X^1_a}{2}, \frac{X^2_b-X^2_a}{2}, \cdots, \frac{X^d_b-X^d_a}{2})$. 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: $K^{X'^1_a,X'^2_a,\cdots,X'^d_a,X'^1_b,X'^2_b,\cdots,X'^d_b}$ = $H(K^{X^1_a,X^2_a,\cdots,X^d_a,X^1_b,X^2_b,\cdots,X^d_b}, \xi_1\xi_2\cdots\xi_d)$, where ξ_i = 0 if X'^i_a = X^i_a and ξ_i = 1 otherwise.

Tables X, IX and VIII show the computation, communication and storage 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

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

TABLE VIII
STORAGE COST

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

TABLE IX
COMMUNICATION COST

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

TABLE X
COMPUTATION COST

acceptably small. In the next section, we show the scalability and efficiency of our protocol over the group key management protocol. Note that x^i denotes the extent of an authorization on the i^{th} domain and (x_{cache}^1, x_{cache}^2, $\cdots$, x_{cache}^d) 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

Qualitative Comparison. In this section we compare our approach with that of a group key management algorithm. In a group key management based approach, one would define the set of users within a d-dimensional bounding box as a group. For example, let us consider a d=1 spatial domain. Suppose a user u_1 subscribes for a 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 KDC has to maintain more groups and group keys (computing and storage cost) as the number of subscribers N increases. The KDC also needs to update active subscribers (like u_1) with new groups and group keys (communication cost) as new users (like u_2) join the system. Additionally, the KDC has to maintain all subscriptions made by all active subscribers in order to define groups and compute the key updates. Our approach allows the KDC to be *stateless* and ensures that the cost of a subscription is small and *independent* of the number of subscribers N. As highlighted in Section II, the stateless nature of our authorization service allows us to distribute and replicate it *on demand* to handle bursty loads.

Analytical Comparison. 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, $\cdots$, x^d) denote the size of a subscription range along the d-dimensions.

Let us suppose that $f_i(s)$ denotes an identical and independently distributed probability density function that a subscriber subscribes for a range $(s, s + x_i)$ in the i^{th}-dimension. Noting the fact two subscriptions $(s, s + x_i)$ and $(r, r + x_i)$ overlap if $s - x_i \leq r \leq s + x_i$, the probability of overlap in the i^{th} dimension is given by op = $\sum_s (f_i(s) * \sum_{r=s-x_i}^{s+x_i} f_i(r))$. For the sake of simplicity let us suppose $x_i \ll X_{max}^i$ such that $f_i(s)$ can be approximated to linear function over the small range $(s - x_i, s + x_i)$. In this case, the probability of overlap could be approximated to op = $2x_i * \sum_s f_i(s)^2$. Given that $\sum_s f_i(s)$ = 1, one can show that op is minimal when $f_i(s)$ = $\frac{1}{X_{max}^i}$ for all s, that is, if $f_i(s)$ follows a uniform and random distribution. Observe that smaller the overlap lower is the cost for group key management protocols.

In the following portion of this section, we assume a uniform and random subscription range distribution, namely, best case scenario for group key management protocols. In this case, the probability of overlap is approximated to $\frac{2x^i}{X_{max}^i}$ (if, $x^i \ll X_{max}^i$). Note that if $x^i \geq \frac{X_{max}^i}{2}$ 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 equation 2. Therefore, the average number of active users whose group keys need to be updated is N*$Pr_{overlap}$.

$$Pr_{overlap} = \frac{\prod_{i=1}^{d}(2x^i)}{\prod_{i=1}^{d}(X_{max}^i)} = 2^d \prod_{i=1}^{d} \frac{x^i}{X_{max}^i} \qquad (2)$$

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 sub-boxes. Figure 2 illustrates the creation of new sub-boxes as new users join the system for a d=2 dimensional domain. The size of the 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 given by Equation 3.

$$cost_{gkm} = 2^d * N * 2^d \prod_{i=1}^{d} \frac{x^i}{X_{max}^i} \qquad (3)$$

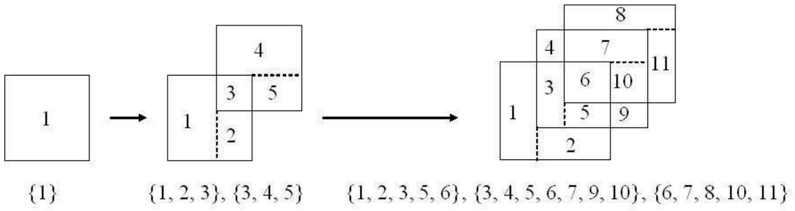

Fig. 2. User Join: Group Key Management

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

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

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

TABLE XII
d=3, N = 10^3

Distribution	Parameter	x
Exponential	$\frac{1}{\lambda}$ = $0.01X_{max}$	2^{800}
Exponential	$\frac{1}{\lambda}$ = $0.1X_{max}$	2^{320}
Exponential	$\frac{1}{\lambda}$ = $0.5X_{max}$	2^{72}
Gaussian	μ = $0.5X_{max}$, σ = $0.01X_{max}$	2^{768}
Gaussian	μ = $0.5X_{max}$, σ = $0.1X_{max}$	2^{477}
Gaussian	μ = $0.5X_{max}$, σ = $0.5X_{max}$	2^{111}
Zipf	γ = 0.01	2^{38}
Zipf	γ = 0.1	2^{88}
Zipf	γ = 0.5	2^{192}

TABLE XIII
d=3, N = 10^2, $\frac{x}{X_{max}}$ = 0.1

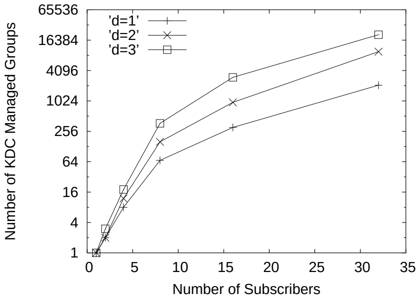

Fig. 3. Scalability Issue with Group Key Management Protocols

The cost of a new user join in our key management protocol is $cost_{stauth}$ = $2^{d-1} * \frac{\sum_{i=1}^{d} \log x^i}{d}$. The ratio of the costs is given by Equation 4.

$$cost_{gkm} : cost_{stauth} = \frac{2^{d+1} * N * d}{\sum_{i=1}^{d} \log x^i} * \prod_{i=1}^{d} \frac{x^i}{X^i_{max}} \tag{4}$$

Let us for the sake of simplicity suppose that the subscription range along each dimension $x^i = x$ and the maximum subscription range along each dimension $X^i_{max} = X_{max}$. Then the ratio becomes $\frac{2^{d+1} * N}{\log x} * \left(\frac{x}{X_{max}}\right)^d$. Now, setting N = 10^4, $d = 3$ and $\frac{x}{X_{max}}$ = 0.1, we observe that $cost_{gkm}$:$cost_{stauth}$ is smaller than one only if $x \geq 2^{160}$. Tables XI and XII show the maximum value of x for d = 3-dimensional domain such that $cost_{stauth} \leq cost_{gkm}$ for different values of N and $\frac{x}{X_{max}}$.

Recall that the uniform and random distribution of the subscription range presents the best case scenario for the group key management approach. However, a realistic scenario wherein a large collection of users share common interests is typically modeled using heavy tailed distributions. Table XIII shows the largest subscription range such that $cost_{stauth}$ $\leq cost_{gkm}$ for three distributions: exponential, Gaussian and Zipf distributions with various parameter values. Note that these distributions are truncated and renormalized to the range $(0, X_{max})$. Observe that as the standard deviation increases, the probability of overlap between two subscription ranges decreases, thereby reducing the cost of the group key management algorithms. On the other hand, our approach is agnostic to the distribution of user interests. Table XIII demonstrates the ability of our approach to handle large and fine grained domains and yet achieve significantly lower costs than the group key management approach.

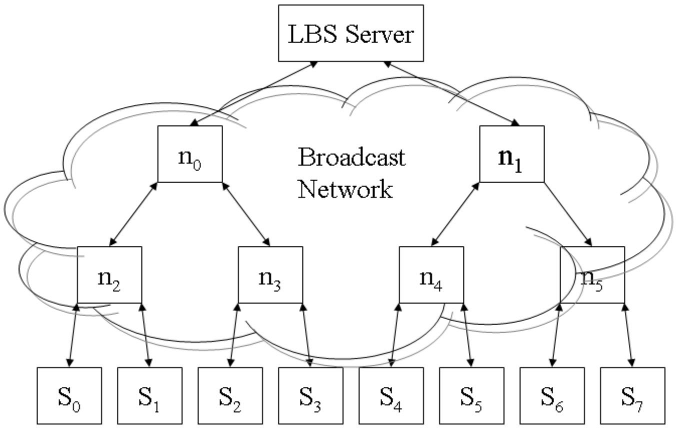

Fig. 4. Siena Broadcast Network: subscriber(S) and network node(n)

IV. Experimental Evaluation

We have implemented our key management algorithms on Siena publish-subscribe network [11]. Siena is a wide-area publish-subscribe network that allows events to be disseminated from a LBS server (publisher) to a geographically scattered group of subscribers. We used GT-ITM [28] topology generator to generate an Internet topology consisting of 63 nodes. The round trip times on these links varied from 24ms to 184ms with mean 74ms and standard deviation 50ms. We constructed a 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-

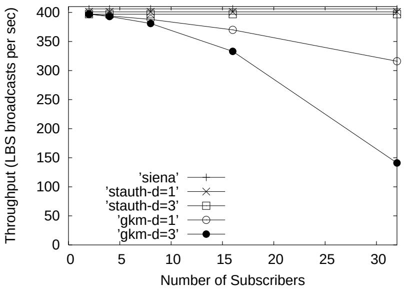

Fig. 5. Throughput Vs Number of Subscribers

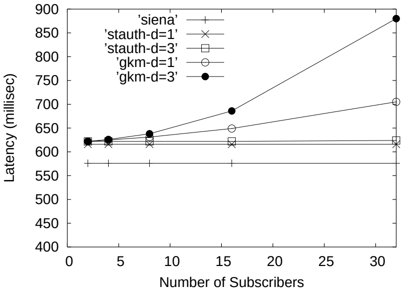

Fig. 6. Latency Vs Number of Subscribers

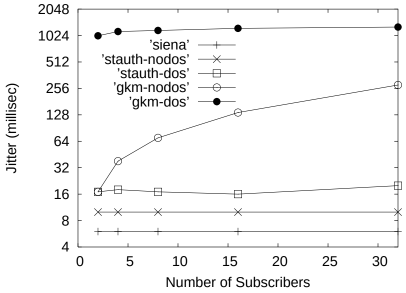

Fig. 7. Resilience to DoS Attacks

ITM topology generator (see figure 4).

All experimental results presented in this section were averaged over 5 independent runs. We simulated a spatial-temporal space of volume 1024 × 1024 × 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 two 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 three experimental results. We first demonstrate the scalability problems in group key management protocols by measuring the number of groups that need to be managed by the KDC. Second, we measure the overhead of our algorithms over the insecure LBS system in terms of its throughput and latency. Third, 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.

Scalability. Figure 3 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 simulation results indicate that even for a modest set of 1000 subscribers the number of managed groups could be about 2^{112}.

Throughput and Latency. Figures 5 and 6 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 to less than 12% of the overhead. Traditional group key management protocols on the other hand incur significant drop in throughput (62.5% for N = 32) and increase in latency as the number of subscribers increase (52% for N = 32). Our simulation results indicate that for N = 1000 subscribers, the throughput could drop is about 99.96% and the increase in latency is about 140 times.

DoS Attack. Figure 7 shows the jitter (standard deviation in inter-packet 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 millisecond, 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 that 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.

V. Related Work

Group key management protocols using a centralized server approach that distributes group keys using unicast was proposed in [15]. Iolus improves the scalability of this approach using distributed hierarchical key servers [18]. Several authors have attempted to use multicast routers to improve the performance of key distribution algorithms [19]. Since then significant amount of work has been done in this field using the concept of a logical key hierarchy [14]. Several papers [4], [23], [24], [26], [17], [9], [10], [21] 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 [21], [27]. A detailed survey along with comparisons amongst various group key management protocols is described in [22].

In this paper we have demonstrated the scalability and performance issues when using group key management protocols with flexible spatial-temporal authorization models and proposed key management algorithms to handle them. Our key management algorithms fall under the category of hierarchical key derivation algorithms [17]. Such algorithms have been commonly used in the field of file systems to support access control graphs [13], [5]. Our approach builds on the hash tree based approach suggested in the MARKS protocol [8] and thus incurs no *leave* cost when a user's authorization expires, and the *join* cost is independent of the number of users in the system.

VI. 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 demonstrates our performance and scalability benefits, while preserving the security guarantees.

References

[1] Garmin. http://www.garmin.com.

[2] Loc aid. http://www.loc-aid.net.

[3] Veripath navigator. http://veripath.us.

[4] K. Aguilera and R. Strom. Efficient atomic broadcast using deterministic merge. In *19th ACM PODC*, 2000.

[5] M. Atallah, K. Frikken, and M. Blanton. Dynamic and efficient key management for access hierarchies. In *ACM CCS*, 2005.

[6] M. J. Atallah, M. Blanton, and K. B. Frikken. Efficient techniques for realizing geo-spatial access control. In *Asia CCS*, 2007.

[7] M. J. Atallah, M. Blanton, and K. B. Frikken. Incorporating temporal capabilities in existing key management schemes. In *ESORICS*, 2007.

[8] B. Briscoe. Marks: Zero side-effect multicast key management using arbitrarily revealed key sequences. In *1st Workshop on Networked Group Comm*, 1999.

[9] R. Canetti, J. Garay, G. Itkis, and D. Micciancio. Multicast security: A taxonomy and some efficient constructions. In *IEEE INFOCOM, Vol. 2, 708-716*, 1999.

[10] R. Canetti, T. Malkin, and K. Nissim. Efficient communication-storage tradeoffs for multicast encryption. In *EUROCRYPT, LNCS, vol. 1599, Springer Verlag, pp: 459-474*, 1999.

[11] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. In *ACM TOCS, 19(3):332-383*, 2001.

[12] E. Eastlake. US secure hash algorithm I. http://www.ietf.org/rfc/rfc3174.txt, 2001.

[13] K. Fu, M. F. Kaashoek, and D. Mazieres. Fast and secure distributed read-only file system. In *Proceedings of the 4th OSDI, pp: 181-196*, 2000.

[14] H. Harney and E. Harder. Logical key hierarchy protocol. http://www.rfc-archive.org/getrfc.php?rfc=2093.

[15] H. Harney and C. Muckenhirn. Group key management protocol (gkmp) architecture. http://www.rfc-archive.org/getrfc.php?rfc=2094.

[16] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-hashing for message authentication. http://www.faqs.org/rfcs/rfc2104.html.

[17] D. A. McGrew and A. T. Sherman. Key establishment in large dynamic groups using one-way function trees. In *Tech. Rep. No. 0755 (May), TIS Labs at Network Associates, Inc., Glenwood, MD*.

[18] S. Mittra. Iolus: A framework for scalable secure multicasting. In *Proceedings of ACM SIGCOMM*, 1997.

[19] R. Molva and A. Pannetrat. Scalable multicast security in dynamic groups. In *6th ACM CCS*, 1999.

[20] NIST. AES: Advanced encryption standard. http://csrc.nist.gov/CryptoToolkit/aes/.

[21] A. Perrig, D. Song, and J. D. Tygar. ELK: A new protocol for efficient large group key distribution. In *Proceedings of IEEE Security and Privacy*, 2001.

[22] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. In *Journal of the ACM Computing Surveys, Vol 35, Issue 3*, 2003.

[23] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner. The versakey framework: Versatile group key management. In *IEEE Journal on Selected Areas in Communications (Special Issue on MiddleWare) 17, 9(Aug), 1614-1631*, 1999.

[24] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. In *RFC 2627*, 1999.

[25] C. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. In *ACM SIGCOMM*, 1998.

[26] C. K. Wong, M. G. Gouda, and S. S. Lam. Secure group communications using key graphs. In *IEEE/ACM Transactions on Networking: 8, 1(Feb), 16-30*, 2000.

[27] C. K. Wong and S. S. Lam. Keystone: A group key management service. In *ICT*, 2000.

[28] E. W. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proceedings of IEEE Infocom*, 1996.