International Journal of Engineering & Technology, 7 (1.3) (2018) 24-27

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Scalable dynamic key management for location based services

S. Ramamani*, S.Karthika

Asst. Prof./CSE, Karpagam College of Engineering, Coimbatore, India
**Corresponding author E-mail: ramatheju23@gmail.com*

Abstract

A Scalable key management for enforcing spatial-quality access control on public broadcast services. This authorization model is used to construct Authorization keys using efficient, secure and scalable Hierarchical key. Secure media broadcast over Internet poses unique security challenges. One important problem for public broadcast location- services (LBS) is to enforce access control on a large number of subscribers. This is achieved by providing an authorization model in constructing authorization keys using efficient, secure and scalable Hierarchical key graphs. And minimizes number of keys that needs to be distributed is thus scalable to a large number of subscribers and the dimensionality of the authorization model. In an offline basis the map viewer is loaded with collection of tiles or segments. The entire map is not loaded fully. If the user key and the segment key is matched with each other then the requested regions are loaded and displayed. Otherwise is blocked for the unauthorized user.

Keywords: *Access Control, Key Management, Location-Based Services (LBS), Scalability And Performance*

1. Introduction

Observe that the group key management server has to not only maintain more keys (computing and storage cost) as the number of subscribers increases, but also update keys at one or more 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

2) User join and leave requires the KDC to broadcast 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 hier-archical key graphs [7], [12] with several features:

1) Number of groups managed by KDC is .

2) User join and leave cost is independent of

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

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.

2. Literature Survey

Key Management For Multicast: Issues And Architectures D.Wallner [1] discussed the difficult problem of key management in multicast communication sessions. It focuses on two main areas of concern with respect to key management, which are, initializ-ing the multicast group with a common net key and rekeying the multicast group. A rekey may be necessary upon the compromise of a user or for other reasons (e.g., periodic rekey).

In particular, this identifies a technique which allows for secure compromise recovery, while also being robust against collusion of excluded users. This is one important feature of multicast key management which has not been addressed in detail by most other multicast key management proposal.

The benefits of this proposed rekeying technique are that it minimizes the number of transmissions required to rekey the multicast group and it imposes minimal storage requirements on the multicast group.

A Scalable Group Re-Keying Approach For Secure Multicast
A novel approach to scalable group re-keying for secure multicast is described [2]. Our approach Kronos is based upon the idea of periodic group re-keying. Their approach are by showing that if a group is re-keyed on each membership change, as the size of the group increases and/or the rate at which members leave and join the group increases, the frequency of re-keying becomes the primary bottleneck for scalable group re-keying.

In contrast, Kronos can scale to handle large and dynamic groups because the frequency of re-keying is independent of the size and membership dynamics of the group. Next, they describe how Kronos can be used in conjunction with distributed key management frameworks such as IGKMP, that use a single group-wide session key for encrypting communications between members of the group.

The operation of the Kronos protocol is similar to that of IGKMP with two key differences. First, the DKD or group manager is not directly involved in generating the new group traffic encryption key that is distributed by the AKDs to the existing members of the group in their area. Second, Kronos uses periodic re-keying to decouple the rate of re- keying from group size and membership dynamics. Under Kronos, group re-keys are not driven by member join or leave requests. Instead, at periodic intervals, all the mem-ber join and leave requests that have accumulated at an AKD are processed and the new multicast traffic encryption key is securely transmitted to the existing members of the group.

ELK, a New Protocol for Efficient Large-Group Key Distribution Secure media broadcast over the Internet poses unique security challenges[3]. One problem access control to a large number of subscribers in a public broadcast. A common solution is to encrypt the broadcast data and to disclose the decryption key to legitimate receivers only. However, how do we securely and efficiently establish a shared secret among the legitimate receivers? And most importantly, how can we efficiently update the group key securely if receivers join or leave? How can we provide reli-ability for key update messages in a way that scales up to large groups?

Current schemes feature efficient key up- date mechanisms assuming that the key updates are communicated reliably to the receivers. In practice, however, the principal impediment to achieve a scal-able system is to distribute the key updates reliably to all receiv-ers. We have designed and implemented ELK, a novel key dis-tribution protocol, to address these challenges with the following features:

•ELK features perfectly reliable, super-efficient member joins.
•ELK uses smaller key update messages than previous protocols.
•ELK features a mechanism that allows short hint messages to be used for key recovery allowing a tradeoff of communication overhead with member computation.

ELK proposes to append a small amount of key update information to data packets, such that the majority of receivers can recover from lost key update messages

HMAC
In cryptography, HMAC (Hash-based Message Authentication Code), is a specific construction[4] for calculating a message authentication code (MAC) involving a cryptographic hash function in combination with a secret key. As with any MAC, it may be used to simultaneously verify both the data integrity and the authenticity of a message. Any cryptographic hash function, such as MD5 or SHA-1, may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-MD5 or HMAC-SHA1 accordingly. The cryptographic strength of the HMAC de-pends upon the cryptographic strength of the underlying hash function, the size of its hash output length in bits and on the size and quality of the cryptographic key.

An iterative hash function breaks up a message into blocks of a fixed size and iterates over them with a compression function. For example, MD5 and SHA-1 operate on 512-bit blocks. The size of the output of HMAC is the same as that of the underlying hash function (128 or 160 bits in the case of MD5 or SHA-1, respect- ively), although it can be truncated if desired.

LSD BROADCAST ENCRYPTION Broad cast Encryption schemes[5] enablea center to broadcast encrypted programs so that only designated subsets of users can decrypt each program. The stateless variant of this problem provides each user with a fixed set of keys which is never updated. The best scheme published so far for this problem is the "subset difference" (SD) technique of Naor Naor and Lotspiech, in which each one of the n users is initially given O(log2(n)) symmetric encryption keys. This allows the broadcaster to define at a later stage any subset of up to r users as "revoked", and to make the program accessible only to their complement by sending This reduces the number of keys given to each user by almost a square root factor without affecting the other parameters. In addition, we show how to use the same LSD keys in order to address any subset defined by a nested combination of inclusion and exclusion conditions with a number of messages which is proportional to the complexity of the description rather than to the size of the subset. The LSD scheme is truly practical, and makes it possible to broadcast an unlimited number of programs to 256,000,000 possible custom-ers by giving each new customer a smart card with one kilobyte of tamper-resistant memory. It is then possible to address any subset defined by t nested inclusion and exclusion conditions by sending less than 4t short messages, and the scheme remains se-cure even if all the other users form an adversarial coalition.

3. Methodology

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 q denotes the spatial bounding box, and denotes quality (of the map in this scenario). We implemented STauth using JavaScripts (AJAX model) that export three in- terfaces: 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 (coordi-nates, quality) derives the decryption key for a given coordinate and quality tuple; and boolean decryptImage (map, key) decrypts the map image using key.

There are three coordinates in Google Maps: tile, pixel, and zoom level. Google Maps divides

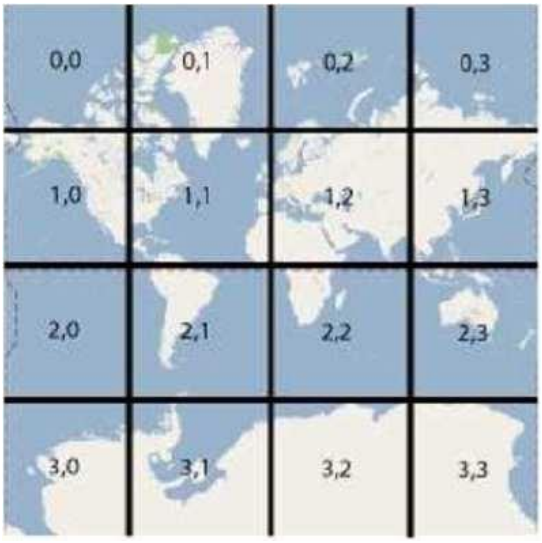

Fig. 3.2: Zoom level 2: 16 tiles.

Google Maps divides Earth into tiles is exactly identical to our approach of defining and partitioning a 2-dimensional bounding box . 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 implemen- tation 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 algo-rithms 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 ren-ders 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.

```
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://hc12.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 3.1:Pseudocode

4. Methodology

System implementation has been done with the help of special map viewer. The Map is loaded in the Map viewer. Then if suppose the User requesting for viewing for a specific location in the map, if the client or user requested is an authorized user then the user can view the specific location what the user requested. Otherwise if the user is an unauthorized user then he is not supposed to view that particular area. And the message which is being broadcasted is also not viewed by the unauthorized user. the authorized user with the key and then he decrypts the key and processed with the key and message. Finally the map is viewed and message is displayed. If the Key is not matched with the Authorized users then the requested region is not viewed and the message is also not displayed.

ALGORITHM FOR THE SYSTEM:

Step 1: The user is given with Location coordinates, Quality and Authorization Box

Step 2: Then Coordinates and Quality is checked for a tile belongs to the Clients spatial authorization box

Step 3: The authorization box consists of keys

Step 4: If the client's key is matched with the authorization box then he is an author-ized user.

Step 5: Authorized user then decrypts the key for the given coordinates, quality.

Step 6: The map with the requested region is displayed

Step 7: If not the unauthorized user requested region is blocked

IMPLEMENTATION DESCRIPTION

The Map viewer consists of collection of tiles of a region like the Google map. The images are seg-mented as tiles with their coordinates. The tiles are loaded with coordinated in the map viewer. The client now request for the region to be viewed and the requested location co-ordinates and the quality along with the authorization box is described. The users are matched with the authorization key and the key is matched with it. If he is an authorized user he views the requested region. Or else the map with the specific region is blocked. And the message broadcasted is not viewed for the unauthorized user.

This is being implemented in an offline basis manner with three modules as Map Loading, Key Authentication, Message Broadcasting. If the key matches with the user and segment only the tiles are loaded and the regions are displayed. Otherwise the message is displayed as an unmatched key and the regions are being blocked.

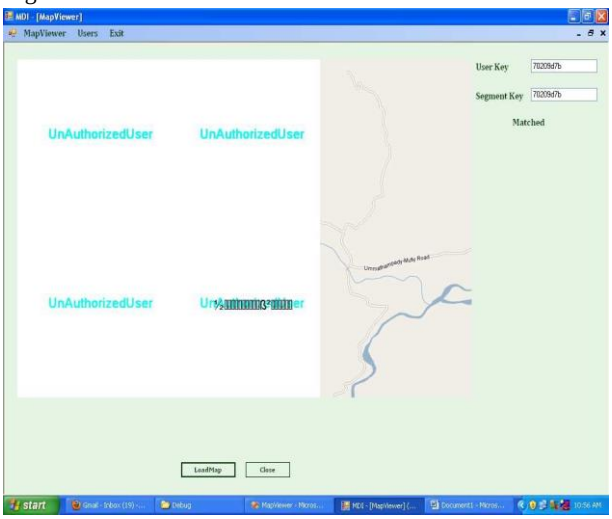

Fig 4.1: Result For unauthorized user

Here the Spatial Quality access control is being implemented. The user gives the Location coordinates and Quality to the server and the co-ordinates is being checked with all other authorization box if the user is matched with.

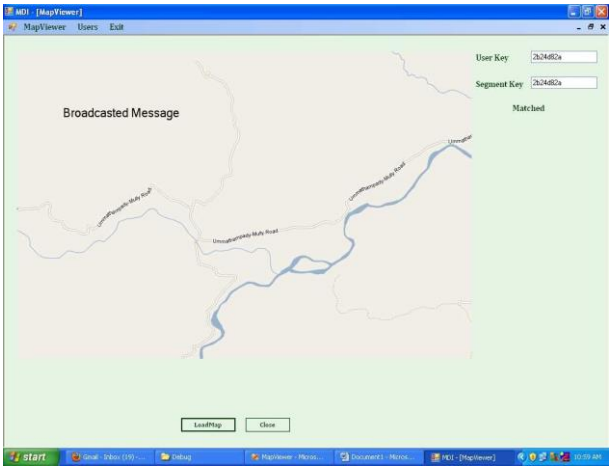

Fig 4.2: Result for Authorised User

5. Conclusion and future work

By loading the map using special viewer and providing the spatial quality access control to confidential areas. The users are checked whether they are authorized users or not. If they are an authorized user then the confidential areas are viewed by the authorized user by maintaining the keys with the authorized and unauthorized users. Then the authorized users can receive the broadcasted message. This is being implemented in Offline manner. Where it is limited to some few applications.

Later as the future work this implementation can be forwarded with the Google map in an Online manner. Such that it can be very much useful and highly secured for various applications.

Rferences

[1] D. Wallner, E. Harder, and R. Agee, (1999) 'Key management for multicast: issues

[2] S. Setia, S. Koussih, S. Jajodia, and E. Harder, (2000) 'Kronos: a scalable group re-keying approach for secure multicast' in IEEE Symposium on Security and Privacy.

[3] M.Atallah.K.Frikken and M.Blanton,(2005) 'Dynamic and efficient key management for access Hierarchies' in Proc.ACM CCsand architectures' IETF RFC 2627.

[4] A.Perrig.D.Song,and J.D.Tygar,(2001)'ELK:Anew protocol for efficient large group key distribution' in IEEE Symposium on Security and Privacy

[5] D.Halevi and A.Shamir,(2002) 'The LSD broadcast encryption scheme' in Proc Crypto.Apache HTTPD server. [Online]. Available: http:// www.apache.org/

[6] Garmin. [Online]. Available: http://www.garmin.com , [3]Google Maps API. [Online]. Available: http://code.google.com/ apis/ maps/

[7] Loc Aid. [Online]. Available: http://www.loc-aid.net [5]Veripath Navigator. [Online]. Available: http://veripath.us [6]K. Aguilera and R. Strom, "Ef cient atomic broadcast using de-terministic merge," in Proc. 19th ACM PODC, 2000, pp. 209–218.

[8] M. Atallah, K. Frikken, and M. Blanton, "Dynamic and ef cient key management for access hierarchies," in Proc. ACM CCS, 2005, pp. 190–202.

[9] M. J. Atallah, M. Blanton, and K. B. Frikken, "Ef cient tech- niques for realizing geo-spatial access control," in Proc. Asia CCS, 2007, pp. 82–92.

[10] M. J. Atallah, M. Blanton, and K. B. Frikken, "Incorporating temporal capabilities in existing key management schemes," in Proc. ESORICS, 2007, pp. 515–530.

[11] 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.

[12] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in Proc. Crypto, 2005, pp. 258–275.

[13] 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

[14] R. Canetti, J. Garay, G. Itkis, and D. Micciancio, "Multicast security: A taxonomy and some ef cient constructions," in Proc. IEEE INFOCOM, 1999, vol. 2, pp.708–716.

[15] R. Canetti, T. Malkin, and K. Nissim, "Ef cient communication-storage tradeoffs for multicast encryption," EUROCRYPT, LNCS, vol. 1599, pp. 459– 474, 1999.