EIJEST

The Egyptian Int. J. of Eng. Sci. and Technology,
Vol.14, No. 1 (Jan. 2011)

BURSTY OPERATION IN SCALABLE KEY MANAGEMENT PROTOCOLS*

Fatma A. Omara *, Sherief I. Zaki ** and Mohamed W. Abo El-soud ***

* Department of Computer Science, Cairo University
** Department of Mathematics, Suez Canal University
*** Department of Computer Science, Suez Canal University

ABSTRACT:

In order to offer secrecy for multicast applications, the traffic encryption key has to be changed whenever a user joins or leaves the system. Such a change has to be communicated to all the current users. The used bandwidth for such re-keying operation could be high when the group size is large. The main proposed protocol is Adaptive Clustering for Scalable Group Key Management (ASGK). According to ASGK protocol, the multicast group is divided into clusters, where each cluster consists of subgroups of members. Each cluster uses its own Traffic Encryption Key (TEK). These clusters are updated periodically depending on the dynamism of the members during the secure session. The modified protocol has been proposed based on ASGK which is called a New Protocol for Scalable Key Management (NPSM). The main challenge of the developed solution is that how to perform a bursty operation in one aggregate operation to reduce the number of re-keying messages for cluster, reduce the frequency of key distributions, increase the scalability of the key distribution protocol and reduce the network traffic.

KEYWORDS: Confidentiality, Group Key Management, Multicast, Subgroup Security Agent, Clusters, Cost Function.

FONCTIONNEMENT RAFALE DANS LA GESTION DES CLÉS EVOLUTIF PROTOCOLES

RESUME :

Afin d'offrir le secret pour des applications multicast, la clé de cryptage du trafic doit être changée chaque fois qu'un utilisateur rejoint ou quitte le système. Un tel changement doit être communiqué à tous les utilisateurs actuels. La bande passante utilisée pour l'opération ressaisie pourrait être élevée lorsque la taille du groupe est grande. Le principal protocole proposé est adaptatif Clustering pour Scalable Groupe de gestion des clés (ASGK). Selon le protocole ASGK, le groupe de multidiffusion est divisé en grappes, où chaque groupe se compose de sous-groupes de membres. Chaque groupe utilise sa propre clé de cryptage du trafic (TEK). Ces groupes sont mis à jour périodiquement en fonction de la dynamique des membres au cours de la session sécurisée. Le protocole modifié a été proposée sur la base ASGK qui est appelé un nouveau protocole de gestion de clés extensible (NPSM). Le principal défi de la solution développée est que la façon d'effectuer une opération en rafale en une seule opération globale visant à réduire le nombre de messages ressaisie pour la grappe, de réduire la fréquence des distributions de clés, d'accroître l'évolutivité du protocole de distribution de clés et de réduire le réseau la circulation.

MOTS-CLÉS: confidentialité, la gestion des clés de groupe, Multicast, sous-groupe Agent de sécurité, Clusters, la fonction de coût.

---

* Received:14/7/2010, accepted: 23/8/2010 (origional Paper)
*** Contact author: (e-mail: drmwagich@hotmail.com)

1. INTRODUCTION

Multicasting is considered an efficient solution for group communication on the Internet ([1]). Instead of sending a separate copy of data per receiver, a sender can send a single copy and the multicast routers in the network make copy and forward packets appropriately to all receivers. Thus, multicasting utilizes network resources such as bandwidth and buffer space efficiently, and reduces load at the sender(s), as well as, the transit routers ([1],[2],[3]).

In order to secure group communications, security mechanisms such as authentication, access control, integrity, and confidentiality are required. Most of these mechanisms rely generally on encryption using one or several keys. The management of these keys, which includes creating, distributing, and updating the keys, constitutes a basic block to build secure group communication applications. Group communication confidentiality requires that only valid users could decrypt the multicast data even if the data is broadcasted to the entire network ([4]).

The confidentiality requirements can be translated further into four key distribution rules ([5]):

- **Non-Group Confidentiality**;
- Users that were never part of the group should not have access to any key that can decrypt any multicast data sent to the group.
- **Forward Confidentiality**;

Users who left the group should not have access to any future key. This ensures that a member cannot decrypt data after leaving the group.

- **Backward Confidentiality**;

A new user that joins a session should not have access to any old key. This ensures that a member cannot decrypt data sent before he joined the group.

- **Collusion Freedom**; Any set of fraudulent users should not be able to deduce the currently used key.

The work in this paper focuses on group key management by using a symmetric cryptosystem such as Advanced Encryption Standard (AES) ([6]). In this system, a symmetric key is used to encrypt data by the source and to decrypt it by the receivers. This key is generally called Traffic Encryption Key (*TEK*).

In order to meet the above requirements, a re-key process should be triggered after each join/leave to or from the secure group. It consists of generating a new *TEK* and distributing it to the members including the new one in case of a join or to the residual members in case of a leave. This process ensures that a new member can not decrypt eventually stored multicast data before its joining and prevents a leaving member from eavesdropping future multicast data.

A critical problem with any re-key technique is scalability; as the re-key process is triggered at each membership change. The number of encryption key update messages may be important in case of frequent join and leave operations, and induces what is commonly called the 1-affects-n phenomenon ([7]). Some solutions have been proposed to organize the group into subgroups with different local traffic encryption keys. This reduces the 1-affects-n impact of the key updating process, but needs decryption and re-encryption operations at

the border of subgroups. These operations may decrease the communication quality.

Challal et al. ([8]) proposed Adaptive Clustering for Scalable Group Key Management (ASGK) that divides the multicast group into subgroups that are managed by Subgroup Security Agents (SSAs). Subgroups are organized into clusters, where all agents in the cluster use the same *TEK*. These clusters are updated periodically by each ASGK agent depending on local dynamism information, i.e. the arrival and leave rate of members. Each agent in addition receives this information from its parent subgroup and computes the re-keying overhead and key translation overhead to decide whether to create a new cluster or to use the *TEK* of its parent agent. The ASGK protocol scales well to large groups by balancing the 1-affects-n overhead and the decryption/ re-encryption operations through the adaptive structures of the clusters depending on the membership dynamism. However, it is noted that ASGK only approximates the 1-affects-n overhead. In particular, the used cost function does not consider the number of affected members.

In this paper, a new protocol called New Protocol for Scalable Key Management (NPSM) has been proposed to tackle the scalability issue by dividing the multicast group into two areas. Each area uses adaptive clustering of encryption. The subgroups are organized into clusters, where each cluster uses the same *TEK*. The partitioning is made in a way that it reduces both re-keying and key translation overheads. The parameters of NPSM introduce more flexibility by giving the possibility to tune the behavior of the overall architecture and its sensitivity to members' dynamism depending on the requirements of the application layer in terms of synchronization and tolerance to jitters in packet delivery. Based on the number of members joining a group and/or leaving a group, group dynamics can be divided according to the following scenarios:

- Single join, single leave,
- Single join and single leave at the same time,
- Multiple joins, multiple leaves, and
- Multiple joins and multiple leaves at the same time.

Where multiple joins and/or multiple leaves are called burst operations.

When the frequency of membership changes is high, the cost of frequent key distributions should be reduced. One feasible way is to accumulate the joins and leaves for a certain period of time such that the frequency of key distributions is reduced ([9]). This can be considered as kind of bursty behavior. Performing a bursty operation in one aggregate operation is important for reducing the number of re-keying messages, reducing the frequency of key distributions, increasing the scalability of the key distribution protocol, and reducing the network traffic. According to the work in this paper, the well-known key-tree key management protocol which described in ([10],[11],[12]) has been extended for secure group communication for the bursty user arrival and departure patterns, especially when multiple joins and multiple leaves occur at the same time. According to the Simulation results, our developed NPSM protocol scales well for large groups by minimizing the 1-affects-n phenomenon,

while it reduces the decryption / re-encryption operations.

The 1-affects-n and the re-encryption overheads are considered relative important when defining an overall cost function for group key management. For a given cost function and a fixed group membership it is then possible to define optimal splits of the group into subgroups. With dynamic group membership, a split may become suboptimal due to membership changes and the subgroups have to be updated to maintain an optimal splitting of the group. This process has overheads of its own.

In the following, a fixed infrastructure of subgroups arranged as a tree is assumed. Each subgroup is managed by a *Subgroup Security Agent (SSA)*. The *SSA* of subgroup A_i is denoted by a_i. A subgroup can be active and use a *TEK* different than that of its parent or passive and use the same *TEK* as its parent. Subgroups that share the same *TEK* form a cluster. The set of subgroups in the same cluster as subgroup A_i are denoted by $C(A_i)$. By this notation we write $a_k \in C(A_i)$ to denote that a_k is the *SSA* of a subgroup A_k in $C(A_i)$.

A *SSA* switches between active and passive state depending on the number of members that have joined the subgroup. The specification of the algorithm that decides which subgroup a new member will join is outside the scope of this paper.

The remaining of this paper is organized as follows: In section 2, an overview of the existed protocol is discussed. In section 3 the Adaptive Clustering for Scalable Group Key Management (ASGK) protocol is discussed. In section 4 the modified cost function is computed. In section 5 the bursty behavior is discussed. In section 6 the bursty cost function of NPSM protocol is computed. The implementation of our NPSM protocol, and the simulation results are in section 7 and ending up by conclusions.

2. A TEK PER SUBGROUP APPROACH

In order to cope with the 1-affects-n problem, approaches have been proposed to organize the multicast group into multiple subgroups. Within each subgroup, a local controller manages a local traffic encryption key. Thereby, any modification in the membership does not affect all the members but only the members of the corresponding subgroup. The IoIus Framework key distribution which has been proposed by Mittra [7] divides the group into regional subgroups, and each subgroup is managed by a trusted Group Security Intermediary (GSI). Each subgroup is treated almost like a separate multicast group with its own subgroup key and its own multicast channel. The GSI in each subgroup manages its subgroup key distribution and authenticates new members joining/ leaving its subgroup. The advantage is that the subgroup runs independently of each other, and the GSI can perform dynamic member operations efficiently and independently without involving members of other subgroups. To bridge data across the subgroups, the GSIs use another separate multicast channel managed by the Group Security Controller (GSC). As a result, each data transmission requires three different multicasts. The sender first multicasts data in its subgroup channel. When the sender's GSI rece-

ives the data, it multicasts it to the other GSIs. Then the other GSIs send multicast data to their subgroup members through their subgroups' multicast channels. Fig.(1) illustrates a hierarchy with six sub-groups. Each of them uses its own TEK.

The TEK per subgroup approach reduces the 1-affects-n problem, which is useful for highly dynamic multicast groups. However, in the case of static multicast groups, this approach requires the decryption and re-encryption of multicast messages whenever they pass from one subgroup to another. Moreover, the decryption/ re-encryption operations induce delays in packet delivery throughout the delivery path. Thereby, the efficiency of this approach depends on the dynamism of the group. Besides, some applications do not tolerated latencies in transmitting data due to decryption/ re-encryption operations. According to the work in this paper, a new proposed solution has been introduced to address the 1-affect-n and re-keying overheads by taking into consideration the dynamic aspect of group membership.

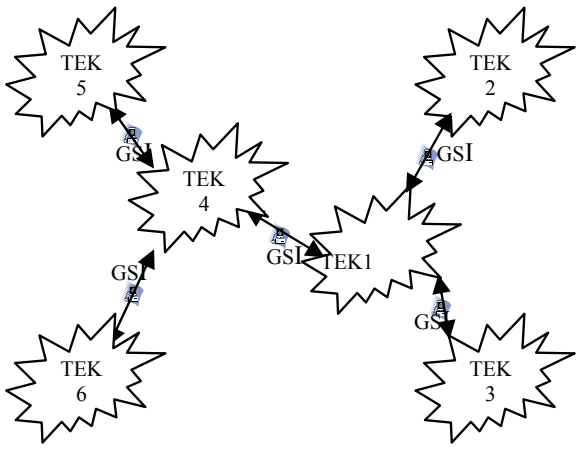

Fig. (1): Example of IoIus architecture

3. THE ASGK PROTOCOL

Challal et al. proposed the Adaptive Clustering for Scalable Group Key Management (ASGK) protocol ([8]), which follows the approach just described. Fig.(2) illustrates the components of the ASGK architecture. ASGK offers an adaptive protocol that maintains good performance during an entire multicast session. The ASGK protocol consists of three phases; *join*, *leave*, and *update* phases.

The multicast communication is denoted with $\Rightarrow$, unicast communication with $\rightarrow$, and out-of-band communication with $\mapsto$. There are multicast channels are assumed from each *SSA* to the members in its subgroup, and a multicast channel between the *SSAs*.

3. 1 Join Phase

When a new member m_{ij} joins in subgroup A_i, all the *SSAs* in the cluster $C(A_i)$ have to distribute a new traffic encryption key *TEK'* to the members in their subgroups. The following four keys will be used in the protocol:

- k_{ij} : a secret key shared between $\mathbf{a_i}$ and m_{ij}.
- ***TEK***: old traffic encryption key.
- ***TEK'***: new traffic encryption key.
- KEK_i: key encryption key shared between a_i and all members in subgroup A_i.

The following protocol is executed when a new member m_{ij} joins in subgroup A_i:

1. $m_{ij} \rightarrow a_i$**:** "join request".
2. $a_i \rightarrow m_{ij} : k_{ij}$.
3. $a_i \rightarrow m_{ij}$**:** $enc(k_{ij}$**;** ***TEK'***, $KEK_i)$.

4. $a_i \Rightarrow \{a_k \mid a_k \in C(A_i)\}$: *"join in subgroup A_i", enc(TEK; TEK').*
5. *For all $a_k \in C(A_i)$; $a_k \Rightarrow A_k$: enc(TEK; TEK').*

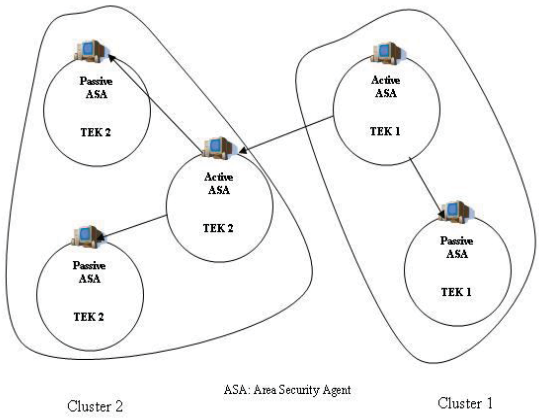

Fig. (2): Adaptive clustering for scalable group key management architecture ([8]).

3. 2 Leave Phase

When member m_{ij} leaves subgroup $\mathbf{A_i}$, all *SSAs* in cluster $C(A_i)$ have to distribute a new *TEK* to the members in their subgroups. The following six keys are used:

- k_{ij}, *TEK, TEK', KEKi* as defined before.
- KEK_i': a new key encryption key for subgroup A_i.
- K_{agt}: a key encryption key shared between all the *SSAs*.

The protocol executed when member m_{ij} leaves subgroup A_i is as follows:

1. $m_{ij} \rightarrow a_i$: *"leave request".*
2. $a_i \rightarrow \{m_{it} \mid m_{it} \neq m_{ij} \in A_i\}$: *enc($k_{it}$; TEK', KEK_i').*
3. $a_i \Rightarrow \{a_k \mid a_k \in C(A_i)\}$: *"leave in subgroup A_i", enc(K_{agt}; TEK').*
4. $a_k \in C(A_i)$; $k \neq i$: $a_k \Rightarrow m_{ik}$: *enc(KEK_k; TEK').*

3. 3 Cluster Update Phase

A distributed cluster update phase is executed periodically taking into account that dynamism distribution over a multicast session is space and time dependent ([13], [14]). Each agent records the arrival and leave rate of members during a given time period. This is the local *dynamism information* λ_i .

In addition, a_i securely receives dynamism information λ_i from its parent subgroup A_j. *SSA* a_i then computes the *1-affects-n* overhead as $\lambda_i + \lambda_j$ and the re-encryption overhead $\rho(A_i)$ as $2.r.Alg_t$ where r is the rate of the multicast traffic and Alg_t the computation time per data unit for encryption taking into consideration the agents' computation power. Let the factor ω indicate the relative importance of the *1-affects-n* overhead in comparison to the re-encryption overhead. Then, a_i takes a local decision to become active or passive depending on the comparison between the weighted overheads (see Fig. (3)).

- If $\omega(\lambda_i + \lambda_j) > \rho(A_i)$ then A_i becomes active and forms a new separate cluster.
- If $\omega(\lambda_i + \lambda_j) <= \rho(A_i)$ then A_i becomes passive and merges with the cluster of its parent.

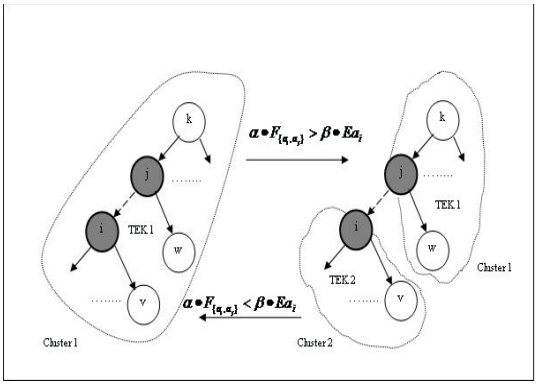

Fig. (3): ASGK protocol

3. 4 Cost Functions

Two overheads are induced by clustering a set of subgroups to use the same *TEK*. The first relates to key translation at the cluster's root agent. This overhead depends on the key translation scheme used. Different schemes have been proposed, such as cipher sequences ([15]), proxy encryption ([16]), and the decryption/re-encryption protocols used in Iolus ([7]) and KHIP ([17]). The Iolus re-encryption overhead in the simulation section will be used as same as ASGK protocol. The second overhead relates to re-keying due to clustering.

The *1-affects-n* overhead can be estimated either by the number of exchanged messages (unicast or multicast) or by the number of affected entities. Table 1 shows the *1-affects-n* overhead for steps 4 and 5 of the ASGK join protocol according to these two approaches. Here, $|C(A_i)|_{subgroups}$ denotes the number of subgroups in cluster $C(A_i)$ and $|C(A_i)|_{members}$ the number of members in cluster $C(A_i)$. Table (2) shows the *1-affects-n* overhead for steps 2, 3, and 4 of the leave protocol.

Table (1): *1-affects-n* Overhead for Join Request

Steps	Messages	Affected Entities
Step 4	1	$\|C(A_i)\|_{subgroups}$
Step 5	$\|C(A_i)\|_{subgroups}$	$\|C(A_i)\|_{members}$

Table (2): *1-affects-n* Overhead for Leave Request

Steps	Messages	Affected Entities
Step 2	$\|A_i\|_{members}$	$\|A_i\|_{members}$
Step 3	1	$\|C(A_i)\|_{subgroups}$
Step 4	$\|C(A_i)\|_{subgroups}$	$\|C(A_i)\|_{members}$

3. 5 Evaluation of the ASGK Protocol

In [8], the disturbance power $dp(A_i)$ of a subgroup A_i is defined as the degree $d(A_i)$ of subgroup Ai in the cluster $C(A_i)$, multiplied by its dynamism information:

$$dp(A_i) = \lambda_i . d(A_i) \qquad (1)$$

The *1-affects-n* overhead $\varphi(C)$ of cluster C is captured by

$$\varphi(C) \approx \sum_{A_j \in C} \lambda_j . d(A_j) \qquad (2)$$

and the cluster's cost function $\gamma(C)$ is defined as

$$\gamma(C) = \omega . \varphi(C) + \rho(C) \qquad (3)$$

where $\rho(C)$ is as above the re-encryption overhead for C and ω a weight factor.

If the number of subgroups is doubled so that membership changes are distributed equitably, it would be moved

from a configuration where all subgroups are active to a configuration where all subgroups become passive forming a single cluster. Note that the ASGK cost function does not consider the overall number of members in a subgroup but only those joining and leaving within the period monitored. As discussed in section 3. 4 the true cost for some of the steps in the join and leave protocols does depend on the full membership.

In summary, the ASGK protocol scales well to large groups by balancing the *1-affects-n* and the re-encryption overheads through the adapting the structure of the clusters depending on membership dynamism. However, its cost function only approximates the *1-affects-n* overhead. In particular, it does not consider the number of affected members. The protocol will be proposed that balance *1-affects-n* and re-encryption overheads when cost is expressed as a function of the number of members affected.

4. NEW COST FUNCTION

Let the re-keying overhead be measured by the number of messages being sent when members join or leave. For measuring the re-keying overhead during a monitoring period *SSA* a_i keeps two counters, a counter τ_i holding the total number of messages it has sent and a counter μ_i holding the number of current subgroup members. The following algorithm is executed during a monitoring period:

- At the start of the period, set $\tau_i \leftarrow 0$, $\gamma_i \leftarrow |C(A_i)|_{subgroups}$.
- When a new member joins, set $\mu_i \leftarrow \mu_i + 1$ and $\tau_i \leftarrow \tau_i + \gamma_i + 1$.
- When a member leaves, set $\mu_i \leftarrow \mu_i - 1$ and $\tau_i \leftarrow \tau_i + \mu_i + \gamma_i + 1$.

It is assumed that first κ_i members join and then λ_i members leave. Then the final value of τ_i is:

$$\tau_i \leftarrow \tau_i + \kappa_i \cdot (\gamma_i + 1) + (\sum_{z=1}^{\lambda_i} \mu_i + \kappa_i - z) + \lambda_i.(\gamma_i + 1)$$

It is assumed that first λ_i members leave and then κ_i members join. Then the final value of τ_i is:

$$\tau_i \leftarrow \tau_i + \kappa_i \cdot (\gamma_i + 1) + (\sum_{z=1}^{\lambda_i} \mu_i - z) + \lambda_i.(\gamma_i + 1)$$

Then, τ_i have two values. The first one is an upper bound and the other is a lower bound.

It is assumed that first $\frac{\lambda_i}{2}$ members leave and then $\frac{\kappa_i}{2}$ members join and then $\frac{\lambda_i}{2}$ members leave and then $\frac{\kappa_i}{2}$ members join. Then the final value of τ_i is:

$$\tau_i \leftarrow \kappa_i \,.\, (\gamma_i + 1) + (\sum_{z=1}^{\lambda_i} \mu_i - \frac{\lambda_i}{2} + \frac{\kappa_i}{2} - z) + \lambda_i.(\gamma_i + 1)$$

Because of the result of this equation is not larger than the first number or not smaller than the second number. Then the two numbers are an upper and a lower bound.

To approximate the cost τ_i we take the centre of this interval and set

$$\mathbf{M(Ai)} = \kappa_i \,.\, (\gamma_i + 1) + (\sum_{z=1}^{\lambda_i} \mu_i + \frac{\kappa_i}{2} - z) + \lambda_i.(\gamma_i + 1)$$

Then, we define the cluster's cost function by

$$\mathbf{Cost(C)} = \omega (\sum_{A_i \in C} M(A_i)) + \rho(\mathbf{C}).$$

Let G ={C1,C2,….Ci,…} with Ci $\cap$ Cj = ϕ if i ≠ j be a set of clusters covering the subgroups of the multicast group. The cost function for the overall system becomes:

$$\text{Cost(G)} = \sum_{C_k \in G} (\omega (\sum_{A_i \in C} M(A_i)) + \rho(\mathbf{C}))$$

Then, the cluster's cost function is based on the true number of affected members in the cluster and the re-encryption overhead.

5. BURSTY BEHAVIOR AND PROPERTIES

Consider an example in ([18]), to illustrate how the aggregate operation reduces the number of keys in subgroup which need to be changed, (see Fig.(4)). Suppose the current members in the subgroups are {m1;m2;m3;m4;m5;m7;m8}, four members {m1;m2;m5;m7} are leaves, while another five members will join at the same time. If a split operation is used, nine separate operations are needed with $\log_2(8) = 3$ keys needing to be changed for each operation. Therefore, the total number of changed keys is 9 * 3 = 27. If all four leaves and five members are joined at the same time (i.e., in one aggregate operation), then the total number of keys needed to be changed is 6 (all internal keys except the parent of m3 and m4) (see Figure 4). 88% (i.e., 1- 6/27) savings in the total number of changed keys is achieved. The reason for the savings is that any shared key is only changed once in an aggregate operation.

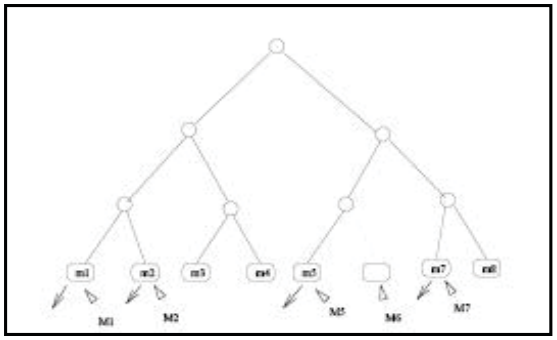

Fig. (4): Bursty behavior: multiple join and leave at the same time

Bursty behavior has some special properties, which can be clarified by means of the following two lemmas (The proofs are omitted because of their obviousness) ([18]). The first lemma states that without considering the keys on leaf nodes, the operations of joining

and leaving are equivalent. The second lemma states that when r joins and s leaves are combined in one aggregate operation, the cost for re-keying min {r; s} keys will be completely saved (covered by max {r; s}).

6. NPSM PROTOCOL

Let the re-keying overhead be measured by the number of messages being sent when members join or leave. For measuring the re-keying overhead during a monitoring period *SSA* a_i keeps four counters, a counter τ_i holding the total number of messages it has sent and a counter μ_i holding the number of current subgroup members and a counter v_i holding multiple joins at the same time and a counter I_i holding multiple leaves at the same time. The following algorithm is executed during a monitoring period:

- At the start of the period, set $\tau_i \leftarrow 0$, $\gamma_i \leftarrow |C(A_i)|_{subgroups}$.
- When a new member joins, set $\mu_i \leftarrow \mu_i + v_i$ **and** $\tau_i \leftarrow \tau_i + \gamma_i + 1$.
- When a member leaves, set $\mu_i \leftarrow \mu_i - I_i$ **and** $\tau_i \leftarrow \tau_i + \mu_i + \gamma_i + 1$.

It is assumed that first η_i times members join and then θ_i times members leave. Then the final value of τ_i is:

$$\tau_i \leftarrow \eta_i \,.\, (\gamma_i + 1) + (\sum_{z=1}^{\lambda_i} \mu_i + \kappa_i - z) + \theta_i \,.\, (\gamma_i + 1)$$

It is assumed that first θ_i times members leave and then η_i times members join. Then the final value of τ_i is:

$$\tau_i \leftarrow \eta_i \,.\, (\gamma_i + 1) + (\sum_{z=1}^{\lambda_i} \mu_i - z) + \theta_i \,.\, (\gamma_i + 1)$$

Then, τ_i have two values. The first one is an upper bound and the other is a lower bound.

To approximate the cost τ_i we take the centre of this interval and set

$$\mathbf{N(Ai)} = \eta_i \,.\, (\gamma_i + 1) + (\sum_{z=1}^{\lambda_i} \mu_i + \frac{\kappa_i}{2} - z) + \theta_i \,.\, (\gamma_i + 1)$$

Then, we define the cluster's cost function by

$$\mathbf{Cost(C)} = \omega (\sum_{A_i \in C} N(A_i)) + \rho\,\mathbf{(C)}.$$

Let G ={C1,C2,….Ci,…} with Ci ∩ Cj = ϕ if i ≠ j be a set of clusters covering the subgroups of the multicast group. The cost function for the overall system becomes:

$$\text{Cost(G)} = \sum_{C_k \in G} (\omega (\sum_{A_i \in C} N(A_i)) + \rho\,\mathbf{(C)})$$

Performing a bursty operation in one aggregate operation is important for reducing the cost function in the NPSM protocol.

If $v_i = 2$, and $I_i = 2$, Then, $\eta_i = \frac{\kappa_i}{2}$, and $\theta_i = \frac{\lambda_i}{2}$. If $v_i = 3$, and $I_i = 3$, Then, $\eta_i = \frac{\kappa_i}{3}$, and $\theta_i = \frac{\lambda_i}{3}$. Thus, $\eta_i \leq \kappa_i$, and $\theta_i \leq \lambda_i$. Here,

$$\eta_i . (\gamma_i + 1) + \theta_i . (\gamma_i + 1) \leq \kappa_i . (\gamma_i + 1) + \lambda_i . (\gamma_i + 1)$$

Then, we find **N(Ai)** ≤ **M(Ai)** and thus our developed NPSM protocol scales well for large groups by minimizing the cost function.

7. SIMULATION RESULTS

For illustration, we measure the number of affected members for NPSM protocol runs when areas form a binary tree with five areas respectively. Our simulations cover a session of three hours where member arrivals follow a Poisson distribution with average inter-arrival time of 20s, and members remain in a session for 30 minutes on average. SSAs execute the cluster update phase every 15 minutes. Figs.(5), and (6) give the result of a simulation using $\omega = 1$ and a rate of multicast traffic of 10 data units per second.

A comparative study has been implemented between the Ancestors protocol and the other protocols using ns2 simulator run sunstation with linux operating system. According to Figure 5 by using the proposed NPSM protocol, the *1-affects-n* overhead is smaller than that the ASGK protocol through the whole update times. According to Figure 6, the proposed Ancestors protocol has the same nearly cost as the ASGK protocol and these protocols are smaller than the Iolus protocol. Generally, the proposed Ancestors protocol is always outperformed the other protocols.

8. DISCUSSION AND CONCLUSIONS

Consider group key distribution to a large and dynamic group. In most applications, some members join and leave any time, these joins and leaves induce re-keying. Changes in group membership require new keys to be distributed. To manage the overhead thus created, a multicast group can be split into subgroups, where subgroups form clusters so that each cluster uses its own Traffic Encryption Key. It is noted that all proposed protocols suffer from great concerns depending on group dynamism where the common *TEK* approaches suffer from the *1-affects-n* phenomenon, where a single group membership (join or leave) changes results in a re-keying process such that all group members have to update the *TEK*.

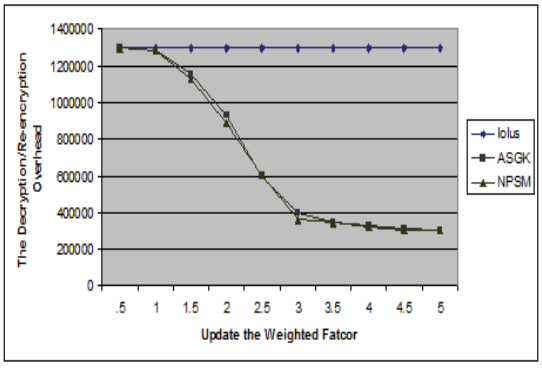

Fig. (5): Comparison of decryption/ re-encryption overhead

Moreover, ASGK protocol relies on dynamic clustering of encryption subgroups depending on the actual

membership dynamism which has been shown to be time and space dependent. But ASGK protocol has disadvantage that the ASGK cost function does not consider the overall number of members in an area but only those joining and leaving within the period monitored. We proposed an adaptive protocol NPSM that relies on dynamic clustering of encryption subgroups depending on the true number of affected members which has been shown to be time and space dependent. In addition to the aggregation of joining and leaving members in secure multicast applications is important for reducing the cost of key distributions. Simulation results show that our scheme achieves better performance trade-offs compared to other schemes in the literature. Therefore, the 1-affects-n phenomenon and hence the re-keying overhead are minimized. The objective of this protocol is to satisfy the balanced between the *1-affects-n* overhead and the decryption/ re-encryption overhead.

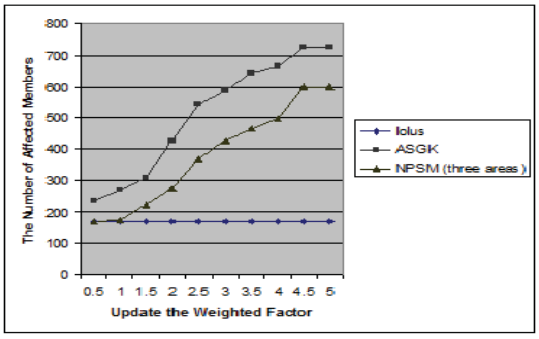

Fig.(6): Comparison of the number of affected members

Future Work

Instead of sending data independently to clients by the server, the peer-to-peer multicast scheme could be used to redistribute the serving load among the clients. This modification will dramatically reduce the server's resource requirement; enable low bandwidth sources to serve high quality live media to up to 100 clients.

REFERENCES

1- S. E. Deering, "Multicast Routing in Internetworks and Extended lANs," ACM SIGCOMM, Aug. 1998.

2- S. Deering, "Host Extensions for IP Multicasting," RFC 1112, 1989.

3- S. E. Deering, "Multicast Routing in a datagram Internetworks," Ph.D. Thesis, Standford University, Dec. 1991.

4- S. Rafaeli; and D. Hutchison, "A Survey of Key Management for Se cure Group Communication," ACM Computing Surveys, Volume 35, Sep.2003, Nr. 3, PP. 309-329.

5- S. Jack; S. Subhash; and V. George, "A Lower Bound for Multicast Key Distribution," IEEE INFOCOM'01 (2001), PP. 1-10.

6- Federal Information Processing Standards Publication, "Advanced Encryption Standard (AES)." (FIPS PUB 197), November 2001.

7- Mittra; and Suvo, "Iolus, A Frame work for Scalable Secure Multicast ing," ACM SIGCOMM (1997), PP.. 1-12.

8- Y. Challal; and Gharout Said; and A. Bouabdallah; and H. Bettahar, "Adaptive Clustering for Scalable Key Management in Dynamic Group Communications," Interna tional Journal of Security and Net works, Volume 3 (2008), Nr. 2, PP. 133-146.

9- R. Chang, D. Engel, D. Kandlur, D. Pendarakis, and D. Saha," Key Mana gement for Secure Internet Multicast Using Boolean Function Minimiza tion Techniques," Proceedings of IN FOCOM'99: Conference on Com puter Communications, PP. 689–698, 1999.

10- G. Caronni, K. Waldvogel, D. Sun, and B. Plattner," Efficient Security for Large and Dynamic Multicast Groups," Proceedings Seventh IEEE International Workshop on Enabling Technologies: Infrastructure for Col laborative Enterprises (WETICE '98) (Cat. No.98TB-100253). Los Alamitos, CA, USA: IEEE, PP. 376–383,1998.

11- G. Noubir, "Multicast Security," European Space Agency, Project: Performance Optimisation of Internet Protocol Via Satellite, April 1998.

12- C. K. Wong, M. Gouda, and S. S. Lam, "Secure Group Communications Using Key Groups," SIG COMM '98, Also University of Texas at Austin, Computer Science Technical report TR 97-23, PP. 68–79, December 1998.

13- K. Almeroth; and M. Ammar, "Collecting and Modeling the Join/Leave Behavior of Multicast Group Members in the Mbone, "Symposium on High Performance Distributed Computing, Syracuse NY (1996), PP. 209-216.

14- K. Almeroth; and M. Ammar, "Multicast Group Behaviour in The Internet's Multicast Backbone (Mbone)," IEEE Communications Magazine, Volume 35 (1997), Nr. 6, PP. 124-129.

15- R. Molva; and A. Pannetrat, "Scalable Multicast Security in Dynamic Groups," 6th ACM Conference on Computer and Communication Security. Singapore (1999), November, PP. 101-112.

16- R. Mukherjee; and J. W.Atwood, "Proxy Encryptions for Secure Multicast Key Management," IEEE Local Computer Networks - LCN'03, Bonn Germany (2003), October, PP. 377-384.

17- C. Shields; and J. J. Garcia-Luna-Aceves, "KHIP-A Scalable Protocol for Secure Multicast Routing," In: ACM SIGCOMM Computer Communications Review, California Univ Santa Cruz Dept of Computer Engineering, Volume 29 (1999), October, Nr. 4, PP. 53-64.

18- X. Zou, B. Ramamurthy, and S. Magliveras (USA)," Efficient Key Management for Secure Group Communications with Bursty Behavior," 2000.