Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

ORGINAL ARTICLE

Ancestors protocol for scalable key management

Dieter Gollmann [a,*], Fatma A. Omara [b], Shrief Ibrahem Zaki [c], Mohamed W. Abo El Soud [d]

[a] *Department of Security, Hamburg-Harburg University, Germany*
[b] *Department of Computer Science, Cairo University, Egypt*
[c] *Department of Mathematics, Suez Canal University, Egypt*
[d] *Department of Computer Science, Suez Canal University, Egypt*

Received November 2009; accepted 13 April 2010
Available online 13 August 2010

KEYWORDS

Confidentiality;
Group key management;
Multicast;
Area security agents;
Clusters;
Cost function

Abstract Group key management is an important functional building block for secure multicast architecture. Thereby, it has been extensively studied in the literature. 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 areas 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 with some modifications to balance the number of affected members and the encryption/decryption overhead with any number of the areas when a member joins or leaves the group. This modified protocol is called *Ancestors* protocol. According to *Ancestors* protocol, every area receives the dynamism of the members from its parents. The main objective of the modified protocol is to reduce the number of affected members during the leaving and joining members, then 1 affects *n* overhead would be reduced. A comparative study has been done between

* Corresponding author.
E-mail addresses: diego@tu-harburg.de (D. Gollmann), F.Omara@fci-cu.pdu.eg (F.A. Omara), sherif.ibrahem@gmail.com (S.I. Zaki), drmwagieh@hotmail.com (M.W. Abo El Soud)

Peer review under responsibility of Faculty of Computers and Information, Cairo University.
doi:10.1016/j.eij.2010.06.002

ASGK protocol and the modified protocol. According to the comparative results, it found that the modified protocol is always outperforming the ASGK protocol.

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].

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 [3].

The confidentiality requirements can be translated further into four key distribution rules [4]:

- *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) [5]. 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 [6]. Some solutions have been proposed to organize the group into areas 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 areas. These operations may decrease the communication quality. Such schemes have to define a mechanism that decides how a group should be divided. Ideally, the result of such division is optimal could be proved. Optimality of course, depends on the chosen cost functions.

Challal et al. [7] proposed Adaptive Clustering for Scalable Group Key Management (ASGK) that divides the multicast group into areas that are managed by Area Security Agents (*ASAs*). Areas 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 area 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 encryption/decryption 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. The Ancestors protocol will be proposed to provide a better balance between encryption/decryption overhead and mutual impact overhead where mutual impact overhead is expressed as a function of the affected number of members.

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 areas. With dynamic group membership, a split may become suboptimal due to membership changes and the areas 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 areas arranged as a tree is assumed. Each area is managed by an Area Security Agent (*ASA*). The *ASA* of area $\boldsymbol{A}_i$ is denoted by $\boldsymbol{a}_i$. An area can be active and use a *TEK* different than that of its parent or passive and use the same *TEK* as its parent. Areas that share the same *TEK* form a cluster. The set of areas in the same cluster as area $\boldsymbol{A}_i$ are denoted by $\boldsymbol{C}(\boldsymbol{A}_i)$. By abuse of notation we write $\boldsymbol{a}_k \in \boldsymbol{C}(\boldsymbol{A}_i)$ to denote that $\boldsymbol{a}_k$ is the *ASA* of an area $\boldsymbol{A}_k$ in $\boldsymbol{C}(\boldsymbol{A}_i)$.

An *ASA* switches between active and passive state depending on the number of members that have joined the area. The specification of the algorithm that decides which area a new member will join is outside the scope of this paper. When designing an algorithm for updating the state of an *ASA*, it is desirable to make decisions based as much as possible on local information.

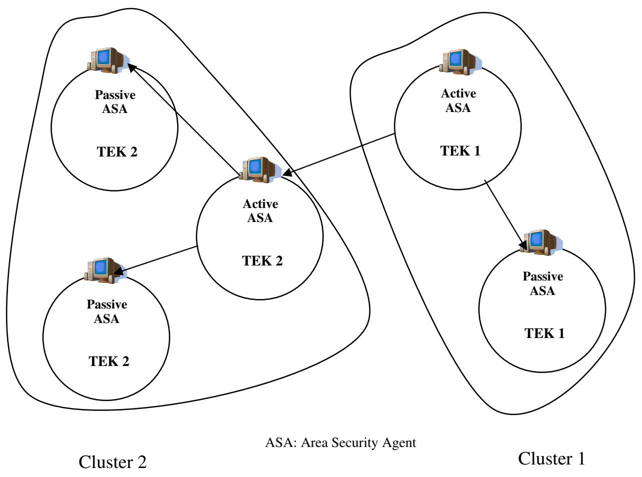

Figure 1 Adaptive Clustering for Scalable Group Key Management Architecture [7].

2. The ASGK protocol

Challal et al. proposed the Adaptive Clustering for Scalable Group Key Management (ASGK) protocol [7], which follows the approach just described. Fig. 1 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 *ASA* to the members in its area, and a multicast channel between the *ASAs*.

2.1. Join phase

When a new member $\boldsymbol{m}_{ij}$ joins in area $\boldsymbol{A}_i$, all the *ASAs* in the cluster $\boldsymbol{C}(\boldsymbol{A}_i)$ have to distribute a new traffic encryption key *TEK′* to the members in their areas. The following four keys will be used in the protocol:

- $\boldsymbol{k}_{ij}$: a secret key shared between $\boldsymbol{a}_i$ and $\boldsymbol{m}_{ij}$.
- ***TEK***: old traffic encryption key.
- ***TEK′***: new traffic encryption key.
- $\boldsymbol{KEK}_i$: key encryption key shared between $\boldsymbol{a}_i$ and all members in area $\boldsymbol{A}_i$.

The following protocol is executed when a new member $\boldsymbol{m}_{ij}$ joins in area $\boldsymbol{A}_i$:

1. $\boldsymbol{m}_{ij} \rightarrow \boldsymbol{a}_i$: "join request".
2. $\boldsymbol{a}_i \rightarrow \boldsymbol{m}_{ij} : \boldsymbol{k}_{ij}$.
3. $\boldsymbol{a}_i \rightarrow \boldsymbol{m}_{ij} : \boldsymbol{enc}(\boldsymbol{k}_{ij}; \boldsymbol{TEK'}, \boldsymbol{KEK}_i)$.
4. $\boldsymbol{a}_i \Rightarrow \{\boldsymbol{a}_k | \boldsymbol{a}_k \in \boldsymbol{C}(\boldsymbol{A}_i)\}$: "***join in area*** $\boldsymbol{A}_i$", $\boldsymbol{enc}(\boldsymbol{TEK}; \boldsymbol{TEK'})$.
5. ***For all*** $\boldsymbol{a}_k \in \boldsymbol{C}(\boldsymbol{A}_i)$; $\boldsymbol{a}_k \Rightarrow \boldsymbol{A}_k$: $\boldsymbol{enc}(\boldsymbol{TEK}; \boldsymbol{TEK'})$.

2.2. Leave phase

When member $\boldsymbol{m}_{ij}$ leaves area $\boldsymbol{A}_i$, all *ASAs* in cluster $\boldsymbol{C}(\boldsymbol{A}_i)$ have to distribute a new *TEK* to the members in their areas. The following six keys are used:

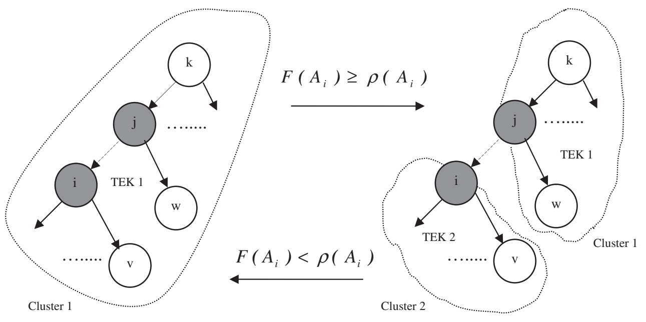

Figure 2 ASGK protocol.

- $\boldsymbol{k}_{ij}$, ***TEK***, ***TEK′***, $\boldsymbol{KEK}_i$ as defined before.
- $\boldsymbol{KEK}'_i$: a new key encryption key for area $\boldsymbol{A}_i$.
- $\boldsymbol{K}_{agt}$: a key encryption key shared between all the *ASAs*.

The protocol executed when member $\boldsymbol{m}_{ij}$ leaves area $\boldsymbol{A}_i$ is as follows:

1. $\boldsymbol{m}_{ij} \rightarrow \boldsymbol{a}_i$: "*leave request*".
2. $\boldsymbol{a}_i \rightarrow \{\boldsymbol{m}_{it} | \boldsymbol{m}_{it} \neq \boldsymbol{m}_{ij} \in \boldsymbol{A}_i\}$: $\boldsymbol{enc}(\boldsymbol{k}_{it}; \boldsymbol{TEK'}, \boldsymbol{KEK}'_i)$.
3. $\boldsymbol{a}_i \Rightarrow \{\boldsymbol{a}_k | \boldsymbol{a}_k \in \boldsymbol{C}(\boldsymbol{A}_i)\}$: "***leave in area*** $\boldsymbol{A}_i$", $\boldsymbol{enc}(\boldsymbol{K}_{agt}; \boldsymbol{TEK'})$.
4. $\boldsymbol{a}_k \in \boldsymbol{C}(\boldsymbol{A}_i)$; $\boldsymbol{k} \neq_i$: $\boldsymbol{a}_k \Rightarrow \boldsymbol{m}_{ik}$: $\boldsymbol{enc}(\boldsymbol{KEK}_k; \boldsymbol{TEK'})$.

2.3. 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 [8,9]. Each agent records the arrival and leave rate of members during a given time period. This is the local *dynamism information* $\boldsymbol{\lambda}_i$.

In addition, $\boldsymbol{a}_i$ securely receives dynamism information $\boldsymbol{\lambda}_i$ from its parent area $\boldsymbol{A}_j$. *ASA* $\boldsymbol{a}_i$ then computes the 1 affects *n* overhead as $\boldsymbol{\lambda}_i + \boldsymbol{\lambda}_j$ and the re-encryption overhead $\boldsymbol{\rho}(\boldsymbol{A}_i)$ as $2 \cdot \boldsymbol{r} \cdot \boldsymbol{Alg}_t$ where $\boldsymbol{r}$ is the rate of the multicast traffic and $\boldsymbol{Alg}_t$ the computation time per data unit for encryption taking into consideration the agents' computation power. Let the factor $\boldsymbol{\omega}$ indicate the relative importance of the 1 affects *n* overhead in comparison to the re-encryption overhead. Then, $\boldsymbol{a}_i$ takes a local decision to become active or passive depending on the comparison between the weighted overheads (see Fig. 2).

- If $\boldsymbol{\omega}(\boldsymbol{\lambda}_i + \boldsymbol{\lambda}_j) > \boldsymbol{\rho}(\boldsymbol{A}_i)$ then $\boldsymbol{A}_i$ becomes active and forms a new separate cluster.
- If $\boldsymbol{\omega}(\boldsymbol{\lambda}_i + \boldsymbol{\lambda}_j) \leqslant \boldsymbol{\rho}(\boldsymbol{A}_i)$ then $\boldsymbol{A}_i$ becomes passive and merges with the cluster of its parent.

2.4. Cost functions

Two overheads are induced by clustering a set of areas 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

Table 1 1 Affects ***n*** overhead for join request.

Steps	Messages	Affected entities
Step 4	1	$\lvert\boldsymbol{C}(\boldsymbol{A}_i)\rvert_{areas}$
Step 5	$\lvert\boldsymbol{C}(\boldsymbol{A}_i)\rvert_{areas}$	$\lvert\boldsymbol{C}(\boldsymbol{A}_i)\rvert_{members}$

Table 2 1 Affects *n* overhead for leave request.

Steps	Messages	Affected entities
Step 2	$\lvert A_i\rvert_{members}$	$\lvert A_i\rvert_{members}$
Step 3	1	$\lvert C(A_i)\rvert_{areas}$
Step 4	$\lvert C(A_i)\rvert_{areas}$	$\lvert C(A_i)\rvert_{members}$

cipher sequences [10], proxy encryption [11], and the decryption/re-encryption protocols used in Iolus [6] and KHIP [12]. 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, $|C(A_i)|_{areas}$ denotes the number of areas 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.

2.5. Evaluation of the ASGK protocol

In [7], the disturbance power $dp(A_i)$ of an area A_i is defined as the degree $d(A_i)$ of area A_i in the cluster $C(A_i)$, multiplied by its dynamism information:

$$dp(A_i) = \lambda_i \cdot d(A_i) \tag{1}$$

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

$$\varphi(C) \approx \sum_{A_j \in C} \lambda_j \cdot d(A_j) \tag{2}$$

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

$$\gamma(C) = \omega \cdot \varphi(C) + \rho(C) \tag{3}$$

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

It is shown in [12] that the update protocol in Section 2.3 is optimal with respect to this cost function.

However, the update function can induce some non-intuitive results. Assume that all areas have the same membership turnover λ. Then, the update condition becomes

$$\omega \cdot \lambda > r \cdot Alg_t \tag{4}$$

for all *ASAs*. Hence, either all areas are active or all areas are passive. Assume further that we have

$$\omega \cdot \lambda > r \cdot Alg_t > \omega \cdot \frac{\lambda}{2} \tag{5}$$

If the number of areas is doubled so that membership changes are distributed equitably, it would be moved from a configuration where all areas are active to a configuration where all areas become passive forming a single cluster. Note 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. As discussed in Section 2.4 the true cost for some of the steps in the join and leave protocols does depend on the full membership.

For illustration, the number of affected members is measured for ASGK protocol runs when areas form a binary tree with seven and 15 areas, respectively. A session of three hours has been simulated where member arrivals follow a Poisson distribution with average inter-arrival time of 20 s, and members remain in a session for 30 min on average. *ASAs* execute the cluster update phase every 15 min. Fig. 3 gives the result of a simulation using $\omega = 1$ and a rate of multicast traffic of 10 data units per second. Observe that the number of affected members for the ASGK protocol with 15 areas is greater than for a configuration with seven areas.

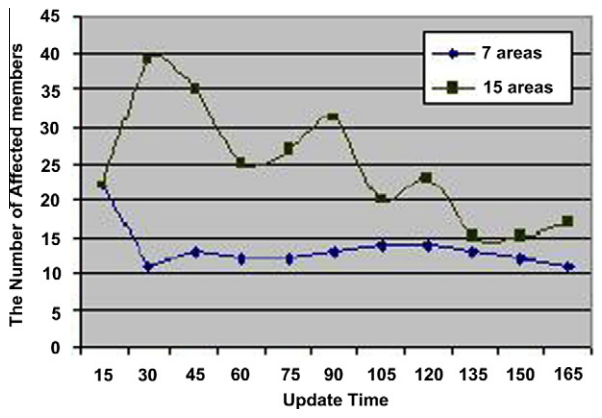

Figure 3 ASGK for different binary trees.

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.

3. Ancestors protocol

The setup is the same as in the ASGK protocol. The multicast group is organized into multiple areas arranged into a tree structure. Each area is managed by an *ASA* which is responsible for the local key management process. An *ASA* can be in two possible states, *active* or *passive*. An *active ASA* uses its own *TEK* for its area and thus has to decrypt and re-encrypt received messages before forwarding them to local members. A *passive ASA* uses the *TEK* of its parent area and hence forwards received messages to local members without decryption/re-encryption. So, the states of the *ASAs* induce a partition of the areas into a set of clusters. Each cluster is composed of a set of areas that share the same *TEK*. The cluster's root *ASA* is active and all internal *ASAs* are passive.

3.1. 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 *ASA* 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 area members. The following algorithm is executed during a monitoring period:

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

3. When a member leaves, set $\boldsymbol{\mu}_i \leftarrow \boldsymbol{\mu}_i - 1$ and $\boldsymbol{\tau}_i \leftarrow \boldsymbol{\tau}_i + \boldsymbol{\mu}_i + \gamma_i + 1$.

It is assumed that first $\boldsymbol{\kappa}_i$ members join and then $\boldsymbol{\lambda}_i$ members leave. Then the final value of $\boldsymbol{\tau}_i$ is:

$$\boldsymbol{\tau}_i \leftarrow \boldsymbol{\tau}_i + \boldsymbol{\kappa}_i \cdot (\gamma_i + 1) + \left(\sum_{z=1}^{\lambda_i} \boldsymbol{\mu}_i + \boldsymbol{\kappa}_i - z\right) + \boldsymbol{\lambda}_i \cdot (\gamma_i + 1)$$

It is assumed that first $\boldsymbol{\lambda}_i$ members leave and then $\boldsymbol{\kappa}_i$ members join. Then the final value of $\boldsymbol{\tau}_i$ is:

$$\boldsymbol{\tau}_i \leftarrow \boldsymbol{\tau}_i + \boldsymbol{\kappa}_i \cdot (\gamma_i + 1) + \left(\sum_{z=1}^{\lambda_i} \boldsymbol{\mu}_i - z\right) + \boldsymbol{\lambda}_i \cdot (\gamma_i + 1)$$

Then, $\boldsymbol{\tau}_i$ have two values. The first one is an upper bound and the other is a lower bound.

Proof. 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 $\boldsymbol{\tau}_i$ is:

$$\boldsymbol{\tau}_i \leftarrow \boldsymbol{\tau}_i + \boldsymbol{\kappa}_i \cdot (\gamma_i + 1) + \left(\sum_{z=1}^{\lambda_i} \boldsymbol{\mu}_i - \frac{\boldsymbol{\lambda}_i}{2} + \frac{\boldsymbol{\kappa}_i}{2} - z\right) + \boldsymbol{\lambda}_i \cdot (\gamma_i + 1) \tag{6}$$

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.

The cluster's cost function, $\boldsymbol{Cost}(\boldsymbol{C}(\boldsymbol{A}_i))$, of a cluster $\boldsymbol{C}(\boldsymbol{A}_i)$ is defined as the number of affected members, $\boldsymbol{\Gamma}(\boldsymbol{C}(\boldsymbol{A}_i))$, and the re-encryption overhead $\boldsymbol{\rho}(\boldsymbol{A}_i)$.

$$\boldsymbol{Cost}(\boldsymbol{C}(\boldsymbol{A}_i)) = \boldsymbol{\omega}\boldsymbol{\Gamma}(\boldsymbol{C}(\boldsymbol{A}_i)) + \boldsymbol{\rho}(\boldsymbol{A}_i) \tag{7}$$

the number of affected members is

$$\boldsymbol{\Gamma}(\boldsymbol{C}(\boldsymbol{A}_i)) = \sum_{\boldsymbol{A}_i, \boldsymbol{A}_j \in \boldsymbol{C}(\boldsymbol{A}_i)}^{\boldsymbol{A}_j \text{ parent of } \boldsymbol{A}_i} \boldsymbol{\Gamma}(\boldsymbol{A}_i, \boldsymbol{A}_j) \tag{8}$$

and

$$\boldsymbol{\Gamma}(\boldsymbol{C}(\boldsymbol{A}_i)) = \sum_{\boldsymbol{A}_m \in \boldsymbol{C}(\boldsymbol{A}_i)} \boldsymbol{\tau}_m \tag{9}$$

Finally,

$$\boldsymbol{Cost}(\boldsymbol{C}(\boldsymbol{A}_i)) = \sum_{\boldsymbol{A}_m \in \boldsymbol{C}(\boldsymbol{A}_i)} \boldsymbol{\tau}_m + \boldsymbol{\rho}(\boldsymbol{A}_i) \tag{10}$$

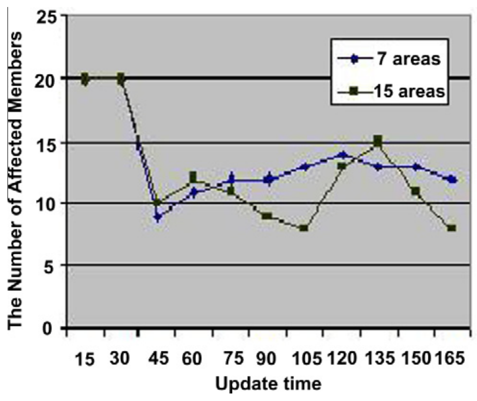

Figure 4 Different binary trees for Ancestors protocol.

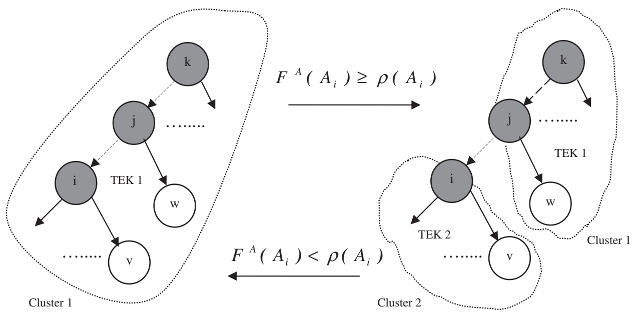

Figure 5 Ancestors protocol.

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

3.2. Evaluation protocol

For illustration, the number of affected members is measured for the Ancestors protocol using a binary tree with seven and 15 areas, respectively. The protocol is implemented by using the some features as in Fig. 3. According to the simulation results, the number of affected members with 15 areas is fewer comparing with seven areas (see Fig. 4). It is found that the number of affected members are balanced using seven and 15 areas (see Fig. 5).

Therefore, the protocol satisfies the balance between mutual impact and re-encryption overheads when cost is expressed as a function of the number of members affected.

4. Cluster update protocol for Ancestors

When the cluster update protocol is executed, every $\boldsymbol{A}_i$ receives the affected members information of areas on the path via passive ancestor areas toward the first active ancestor area, i.e. to the cluster head responsible for generating the cluster *TEK*. We denote the set of Ancestors of area $\boldsymbol{A}_i$ by $\boldsymbol{A}(\boldsymbol{A}_i)$. Then $\boldsymbol{A}_i$ computes the 1 affects *n* overhead $\boldsymbol{F}^A(\boldsymbol{A}_i)$ by:

$$\boldsymbol{F}^A(\boldsymbol{A}_i) = \sum_{\boldsymbol{A}_k \in \boldsymbol{A}(\boldsymbol{A}_i)} \boldsymbol{\tau}_k \tag{11}$$

The re-encryption overhead $\boldsymbol{\rho}(\boldsymbol{A}_i)$ is defined as in Section 2.3. Then, $\boldsymbol{A}_i$ becomes active or passive depending on the comparison between these two overheads (see Fig. 5).

- If $\boldsymbol{F}^A(\boldsymbol{A}_i) > \boldsymbol{\rho}(\boldsymbol{A}_i)$ then $\boldsymbol{A}_i$ becomes active and forms a new separate cluster.
- If $\boldsymbol{F}^A(\boldsymbol{A}_i) \leqslant \boldsymbol{\rho}(\boldsymbol{A}_i)$ then $\boldsymbol{A}_i$ becomes passive and merges with the cluster of its parent.

According to the Ancestor protocol, each agent takes a local decision relying on the affected members information of the ancestor areas in the cluster.

5. Proof of optimality

Definition 1. Let x and y be area security agents, with x parent of y. If $\boldsymbol{\rho}(\boldsymbol{y}) \leqslant \boldsymbol{\Gamma}(\boldsymbol{x}, \boldsymbol{y})$, then if y takes the decision to become active this will decrease the total cost of the partition [12].

Definition 2. Let x and y be area security agents, with x parent of y. If $\boldsymbol{\Gamma}(\boldsymbol{x},\boldsymbol{y}) \leqslant \boldsymbol{\rho}(\boldsymbol{y})$, then if y takes the decision to become passive this will decrease the total cost of the partition [8].

It is supposed when all of the *ASAs* take their decisions, the resulting partition will be the optimal solution. To proof of optimality, the total cost induced by our Ancestors protocol partitioning is denoted by ***Cost*(*New*)**. The total cost induced by any other partition is denoted by ***Cost*(*Other*)**. The areas that make the difference are denoted by $\zeta_a \cup \zeta_p$ where $\zeta_p = \{\boldsymbol{A}_{p1}; \boldsymbol{A}_{p2}; \ldots\}$ are the passive agents in the other partition which are active in the proposed protocol, and $\zeta_a = \{\boldsymbol{A}_{a1}; \boldsymbol{A}_{a2}; \ldots\}$ are the active agents in the other partition which are passive in the proposed protocol. According to the cost function in Eq. (7), we follow that:

$$\begin{aligned}\boldsymbol{Cost}(\boldsymbol{Other}) - \boldsymbol{Cost}(\boldsymbol{New}) &= \sum_{\boldsymbol{A}_{pi} \in \boldsymbol{zeta}_p}^{\boldsymbol{A}_k \text{ parent of } \boldsymbol{A}_{pi}} (\boldsymbol{\Gamma}(\boldsymbol{A}_{pi}, \boldsymbol{A}_k) - \boldsymbol{\rho}\boldsymbol{A}_{pi})) \\ &\quad + \sum_{\boldsymbol{A}_{ai} \in \boldsymbol{zeta}_a}^{\boldsymbol{A}_k \text{ parent of } \boldsymbol{A}_{ai}} (\boldsymbol{\rho}(\boldsymbol{A}_{ai}) - \boldsymbol{\Gamma}(\boldsymbol{A}_{ai}, \boldsymbol{A}_k)) \end{aligned} \tag{12}$$

According to Definition 1 the quantity:

$$\sum_{\boldsymbol{A}_{pi} \in \boldsymbol{zeta}_p}^{\boldsymbol{A}_k \text{ parent of } \boldsymbol{A}_{pi}} (\boldsymbol{\Gamma}(\boldsymbol{A}_{pi}, \boldsymbol{A}_k) - \boldsymbol{\rho}(\boldsymbol{A}_{pi})) \tag{13}$$

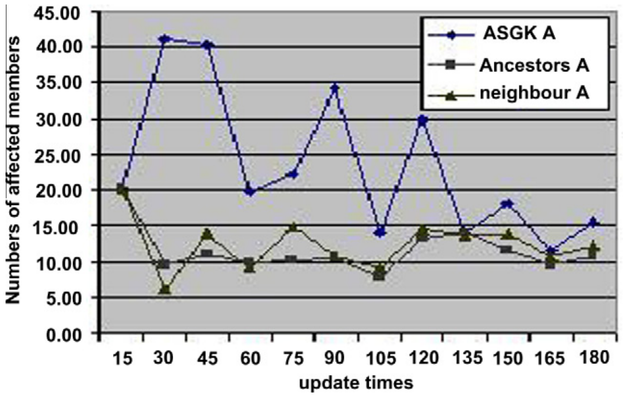

Figure 6 Comparison of 1 affects n overhead.

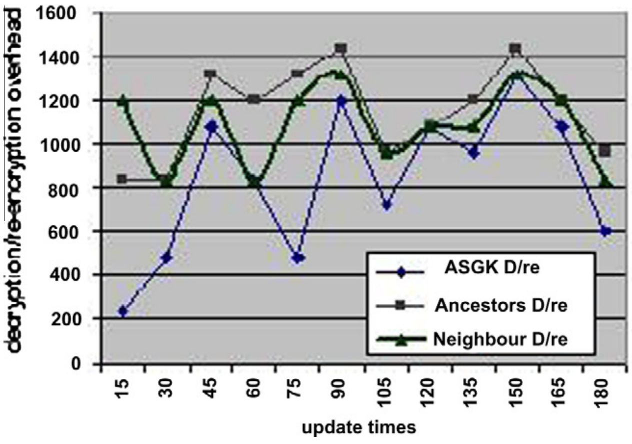

Figure 7 Comparison of cost.

is positive, and according to Definition 2, the quantity:

$$\sum_{\boldsymbol{A}_{ai} \in \boldsymbol{zeta}_a}^{\boldsymbol{A}_k \text{ parent of } \boldsymbol{A}_{ai}} (\boldsymbol{\rho}(\boldsymbol{A}_{ai}) - \boldsymbol{\Gamma}(\boldsymbol{A}_{ai}, \boldsymbol{A}_k)) \tag{14}$$

is positive also. This means that

$$\boldsymbol{Cost}(\boldsymbol{New}) \leqslant \boldsymbol{Cost}(\boldsymbol{Other}) \tag{15}$$

and hence the proposed protocol partition is an optimal solution. Then, the Ancestors protocol outperformed the ASGK protocol.

6. Simulation results

A comparative study has been implemented between the Ancestors protocol and ASGK protocol using ns2 simulator run sunstation with linux operating system. According to Fig. 6 by using the proposed Ancestors protocol, the 1 affects n overhead is smaller than that the ASGK protocol through the whole update times. According to Fig. 7, the proposed Ancestors protocol has the same nearly cost as the ASGK protocol. Generally, the proposed Ancestors protocol is always outperformed the ASGK protocol.

7. Conclusion and future work

7.1. Conclusion

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 areas, where areas 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*.

Moreover, ASGK protocol relies on dynamic clustering of encryption areas depending on the actual membership dynamism which has been shown to be time and space dependent. But ASGK protocol has disadvantage that the number of affected members will grow with the number of areas. ASGK protocol is modified by introducing Ancestors protocol. In Ancestors protocol, each agent receives up to the number of affected members of areas of the path from passive parent's area to active parent's area. The objective of this protocol is to satisfy the balanced between the 1 affects n overhead and the encryption/decryption overhead. According to the simulated results, it is found that our Ancestors protocol is optimal solution.

7.2. 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] Deering S. Multicast routing in internetworks and extended LANs. In: ACM SIGCOMM, vol. 10; August 1998. p. 75–112.

[2] Deering S. Host extensions for IP multicasting. RFC 1112; 1989.

[3] Rafaeli S, Hutchison D. A survey of key management for secure group communication. ACM Comput Surveys 2003;35(3):309–29.

[4] Snoeyink J, Subhash S, Vorghese G. A lower bound for multicast key distribution. In: IEEE INFOCOM'01; 2001. p. 1–10.

[5] Federal Information Processing Standards Publication, Advanced Encryption Standard (AES). FIPS PUB 197; November 2001.

[6] Mittra S. Iolus: a framework for scalable secure multicasting. In: ACM SIGCOMM; 1997. p. 1–12.

[7] Challal Y, Said G, Bouabdallah A, Bettahar H. Adaptive clustering for scalable key management in dynamic group communications. Int J Security Networks 2008;3(2):133–46.

[8] Almeroth K, Ammar M. Collecting and modeling the join/leave behavior of multicast group members in the Mbone. In: Symposium on high performance distributed computing, Syracuse NY; 1996. p. 209–216.

[9] Almeroth K, Ammar M. Multicast group behaviour in the Internet's multicast backbone (Mbone). IEEE Commun Mag 1997;35(6):124–9.

[10] Molva R, Pannetrat A. Scalable multicast security in dynamic groups. In: 6th ACM conference on computer and communication security, Singapore; November 1999. p. 101–112.

[11] Mukherjee R, Atwood JW. Proxy encryptions for secure multicast key management. In: IEEE local computer networks – LCN'03, Bonn Germany; October 2003. p. 377–384

[12] Shields C, Garcia J. KHIP – a scalable protocol for secure multicast routing. ACM SIGCOMM Comput Commun Rev 1999;29(4):53–64.