Asian Journal of Information Technology 15 (18): 3562-3576, 2016
ISSN: 1682-3915

ASAKE: Adaptive Secure Authenticated Key Establishment Mechanism for Dynamic Networks

[1]A. John Prakash and [2]B. Lydia Elizabeth
[1]Ramanujan Computing Centre,
[2]Department of Information Technology, Anna University, 600025 Chennai, Tamil Nadu, India

Abstract: The process of rekeying is required to preserve forward and backward secrecy which induces computational and communication overhead. In dynamic environments, the members move from one position to another or frequent join/leave in the multicast service increases the key management overhead (rekey). When the members become more dynamic, the rate of rekeying increases and hence the overhead increases. This study proposes an Adaptive Secure Authenticated Key Establishment (ASAKE) Mechanism which reduces the effect of state-full member expectation, out-of-sync problem and overhead per rekeying. Also, any change or dynamism in the multicast service group membership results in rekeying and hence in overheads. ASAKE reduces the frequency of rekeying by using a multicrypt and predictive rekeying mechanism and introduces an novel adaptive architecture which keeps the trade-off between 1-affects-n problem and key translation overhead in balance. The performance analysis of the proposed ASAKEshows reduced effect of mobility on the key management protocol. The performance analysis also show that ASAKE is highly scalable and the rekey overhead saved by ASAKE is significant over other standard key establishment protocols.

Key words: Dynamic networks, cryptography, key management, group communication, multicast security, multicrypt

INTRODUCTION

Transmission of datagrams to a group of hosts identified by a single destination address is multicasting. Multicast is intended for group-oriented computing applications such as rescue teams, scientists with sharing of data and requirements for audio and video conferencing. The communication costs like bandwidth consumption and computational costs like sender/router processing for applications that send the same data to multiple recipients is reduced with the use of multicasting. Maintaining the security of multicast packets while delivering to all members with increasingly mobile members is challenging even in wired networks (Hardjono and Tsudik, 2000; Canetti *et al*., 1999; Daghighi *et al*., 2015). Dynamicnet works like internet of things, wireless sensor networks and clouds will have dynamiccapacity links. Many dynamic network environments require the security protocols to be lightweight not only to conserve bandwidth but alsoto prolong node lifetime by reducing the energy consumption (Hegland *et al*., 2006). Hence, the key management protocol should also adapt to the dynamism.

All multicast group members should have a group key to decrypt the datagram sent by the multicast server or source which requires renewal for every membership view change to preserve forward and backward secrecy. Re-keying is the process wherein a key distribution centre (namely, Core) generates and distributes the cryptographic key during each membership event like member join or leave or transfer. Thus, the dynamic re-keying or synchronous re-keying mechanism introduces significant computational and communication overheads.

Motivation: 1-affects-n is the problem when a single change in the multicast group membership affects the n members of the multicast group (Rafaeli and Hutchinson, 2003; Setia *et al*., 2000; Wong *et al*., 2000). When all the members belong to one group then a single member join, leave or change in membership view results in the change of group key for all the n-1 members of the group. To reduce the effect of 1-affects-n problemthe multicast group is sub-divided into small subgroups but another problem called key translation (Setia *et al*., 2000) arises due to the sub-division.

When the multicast group is divided into many small independent subgroups and each subgroup controlled by a Core, the source then sends the data to each subgroup's Core which in turn decrypts the data and

Corresponding Author: A. John Prakash, Ramanujan Computing Centre, Anna University, 600025 Chennai, Tamil Nadu, India

re-encrypts the data with its subgroup key before distributing it to its members. Thus, the decryption/re-encryption or key translation overhead increases when the number of the subgroups increases. Hence, it can be inferred that when the number of concurrent members in the multicast group increases, the number of subgroups has to be increased to reduce the effect of 1-affects-n problem. Similarly, when the overhead due to key translation increases, the number of subgroups has to be decreased. The increasing and decreasing of subgroups has to be dynamic based on the member dynamism so that trade-off between 1-affects-n problem and key translation problem is balanced. Very few key establishment protocols adapt the rekeying process to member dynamism.

Moreover, in mobile environments the nodes move from one transmission range to another within the system. In case a member leaves a subgroup due to mobility and joins another, the old subgroup should rekey to preserve forward confidentiality and the new subgroup should rekey to preserve backward confidentiality resulting in increase of overheads though they are within the group (Mehdizadeh *et al*., 2014). An increase in the number of subgroup would increase the number of member transfers and hence the number of rekeying. Therefore, the rekey strategy should have an efficient member transfer procedure to reduce the overheads caused due to internal movement of nodes.

In key management schemes using symmetric cryptography, more nodes hold the same (group) key increasing the risk of compromise. Furthermore, the symmetric schemes expect state-full members. Out-of-Sync problem also arises if a member misses a rekey message which eventually will exclude that member from the service temporarily. This necessitates the decoupling of the group dynamics and overheads through rekeying. An increased frequency of rekeying increases the effect of out-of-sync problem as well. The overheads and service interruption time increases with the increase in frequency of rekeying which in turn increases with the member dynamism. Thus, it can be inferred that the rekey strategy should minimise the reaction of key establishment protocol to mobility.

Contributions: This paper proposes an Adaptive Secure Authenticated Key Establishment protocol (ASAKE). ASAKE incorporates the following mechanisms:

- Predictive rekeying strategy to reduce the frequency of rekeying. Also to reduce state-full members expectation, out-of-sync problem and effect of coupling between dynamism and rekeying
- Novel Adaptive group strategy which varies the number of subgroups with respect to the change in membership view to balance 1-affects-n and key translation problem
- Novel Member transfer algorithm with less overhead per rekeying and less number of rekeying

MATERIALS AND METHODS

Encryption mechanism overview: In order to preserve forward and backward confidentiality cryptographic keys are required. The cryptographic keys are of two types, namely, Traffic Encryption Keys (TEK) and Key Encryption Keys (TEK). ASAKE's TEK is generated using Multicrypt (Prakash and Uthariaraj, 2009; Arockiasamy *et al*., 2012) and TEK could be of any standard cryptosystems. Multicrypt Cryptosy stemoperates with multiple keys like asymmetric cryptosystems but provides mechanism for dynamic member revocation and addition which helps to achieve the motivations of ASAKE. Multicrypt Cryptosystem is briefly described here and a detailed description can be found in Prakash and Uthariaraj (2009) and Arockiaswamy *et al*. (2012). The multicrypt Cryptosystem, denoted by M = (K, R, ϵ, D) consists of the following procedures:

Key generation (K): The process of key generation involves generation of Hadamard matrix (Harkins *et al*., 2005). Let V be the generated Hadamard matrix given by $V = \{v_1, \ldots, v_N\}$ where $v_1, \ldots, v_N$ are rows of V. Since, V is a Hadamard matrix $v_1, \ldots, v_N$ are mutually orthogonal. Multicrypt uses two keys, namely master key and sub-keys. The master key denoted by K is computed by the Core and private to the Core. The sub-key denoted Γ_i is computed mutually between Core and ith user through an authentication procedureand is private between them.

Authentication procedure (R): It is a probabilistic algorithm to compute the secret initialization of data for a new user subscribing to the system. The authentication procedure R, receives as input N_i and N_c, random nonce associated with the member i and Core, respectively. Let $\Gamma = (\Gamma_1, \Gamma_2, \Gamma_n)$ be the set of sub keys, q be a large prime number, g be the primitive root of q, $Q = \{0, 1, \ldots, q-1\}$ and $N_i, N_c<q$. Throughout this paper, $g^N \bmod q$ is denoted by g^N for simplicity. A typical Diffie-Hellman group setup is assumed (Sterner *et al*., 1996). The authentication procedure then returns ith user's sub-key or secret key given by $\Gamma_i = v_i \times g(N_i \; N \; C) | v_i \in V$. The secure two party's

secret sharing algorithm described in Arockiasamy *et al.* (2012) is assumed for secure communication for computing Γ_i. Then, the master key (K) which is the KEK is the sum of the sub-keys (Γ_i) given by $K = KEK = \Gamma_1 + \Gamma_2 +, \ldots, +\Gamma_n$.

Encryption (ϵ): The encryption procedure ($c \leftarrow \epsilon_K$ (msg)) can be breifly given by:

$$c = K \times msg + v_j \times (g^{sy} \times r) =$$
$$\left((v_1 \times g^{x_1 s} + \ldots + v_{n-1} \times g^{(x_{n-1} s)}\right) \times w + v_j \times r$$

Where:

$v_i \times g^{x_i s} = \Gamma_i\ (s, y, r) \in G$ = The sub key of user i
x_i = The random nonce generated by user i during R
s = THE random nonce generated by $Core_i$ during R

The Optimal Asymmetric Encryption Padding (OAEP) encoding is the standard padding technique (Chan and Chan, 2002) used in this scheme.

Decryption (D): The operation of executing D on Γ_i and c is denoted as $msg \leftarrow D_{ri}$ (c). To decrypt c with decryption function D_{r1} the procedure is briefly described:

$$msg = (c.v_i^T) \times KR_{mi}^{-1}$$

Where:

Kr_{mi}^{-1} = The multiplicative inverse of $g_i^{x.s}$ in group Q
v_i^T = The transpose of the vector vi
$(c.v_i^T)$ = The vector dot product

Key Generation K and Authentication Procedure R are executed by the Core. The Multicrypt encryption mechanism has a property that any member can be dynamically revoked with trivial computations (Prakash and Uthariaraj, 2009; Arockiasamy *et al.*, 2012).

Asake protocol: The objective of Adaptive Secure Authenticated Key Establishment (ASAKE) protocol is to adapt the working of the key establishment with respect to the member dynamics. The overview of ASAKE protocol's working is described succinctly here and each component is explained in detail subsequently.

Architecture overview: The architecture of ASAKE can be associated to a tree structure:

$$T = (Z, V)$$

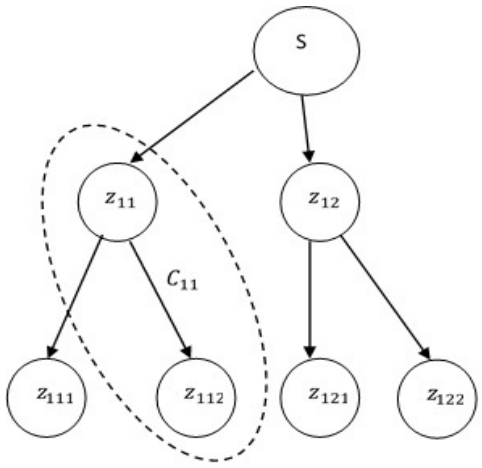

Fig. 1: ASAKE architecture

Where:

Z = The set of sub-trees
V = The set of edges that identify parental relations between sub-tree
$T_i = (Z_i, V_i)$ = A sub-tree of T if T_i is a tree such that $Z_i \subset Z$, $Z_i \neq \phi$ and $V_i \subset V$

Let $G = \{T_1, T_2, \ldots, T_i \ldots\}$ be the family of all possible sub-trees of T. These sub-trees constitute the clusters {Z1, Z2, …, Zi…} of the given hierarchy. A sub-tree is equivalent to a sub-group with Core and a cluster of sub-trees is a combination of sub-groups. Fig. 1 illustrates the tree structure associated to the hierarchy.

Protocol overview: The overview of the working of ASAKE protocol is shown in Fig. 2 which is described step by stepas follows.

The multicast session is started with an initial group formed withthe members wishing to join the multicast session. The average dwell time of mobile members in a subgroup, the average number of members moving to the neighbouring subgroups and arrival rate in the subgroup are computed by the impact prediction mechanism (described in section 3.3) every θ time period.

After every θ time period, each subgroup decides either to merge or split based upon the recommendation of merge/split decision. If a particular subgroup becomes more dynamic it will eventually be split into two subgroups, minimising the 1-affects-n problem. When the split subgroup becomes stable, it will be merged back minimising the key translation or decryption/re-encryption overhead. The rekeying process takes place only after the merge/split operation in the newly adapted architecture.

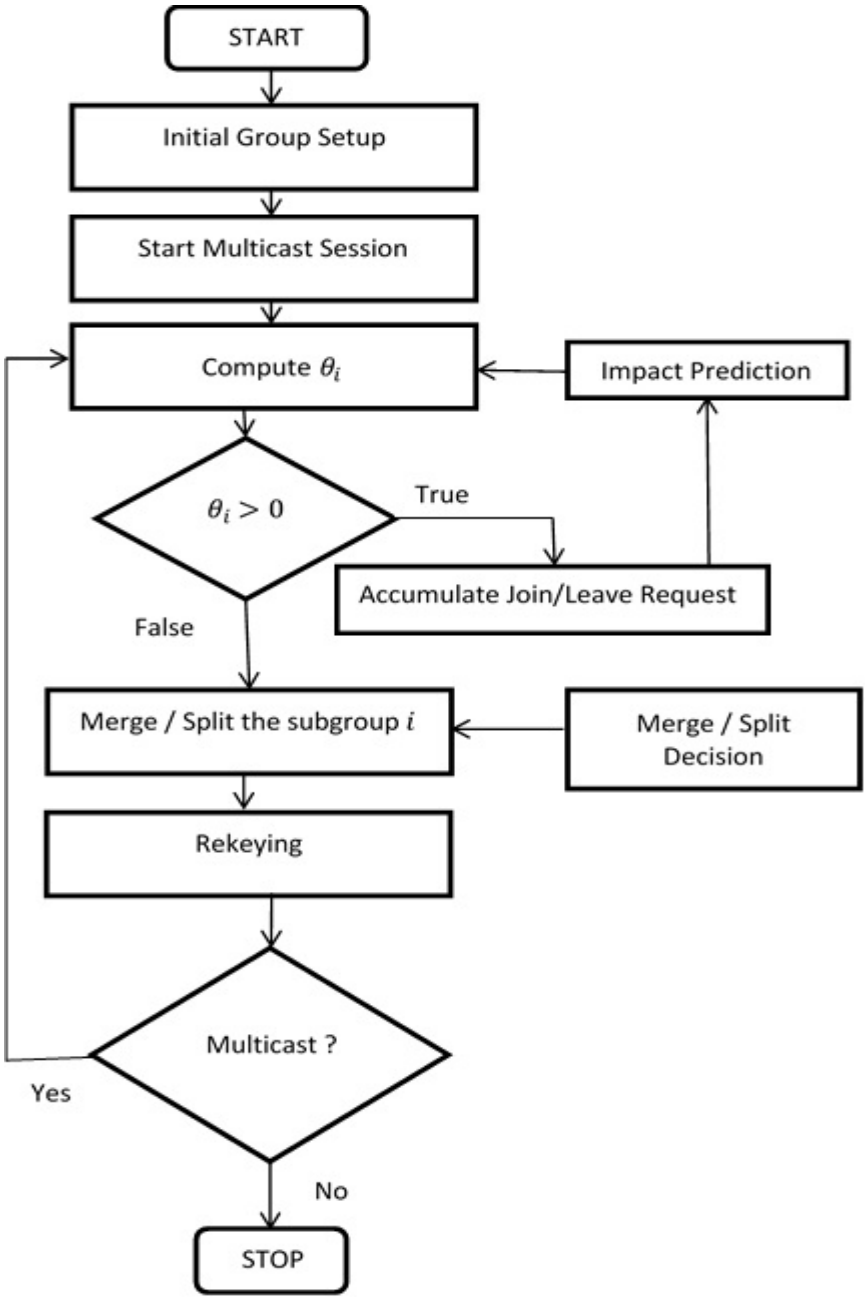

Fig. 2: ASAKE protocol overview

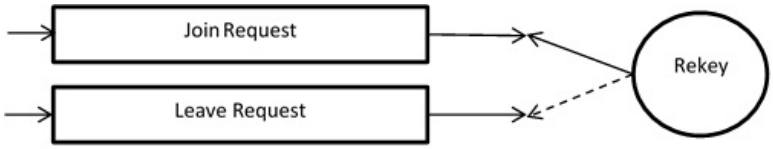

Fig 3: Request Accumulation Queue (RAQ)

Figure 3 shows the Requests Accumulation Queue (RAQ) where the members' join, leave, transfer requests are queued. The join requests (including transfer join) are put on the join requests queue and leave requests (including

transfer leave) on the leave request queue. The predictive rekey mechanism determines the type of rekey mechanism based on the request in the queue that needs to be performed on the predicted rekey time.

When the rekey interval time expires, the subgroup is rekeyed. This process continues as long as the multicast session is in progress. The merge/split of sub-groups or rekeying operations takes place only after every θ time period and hence the frequency of such operations is controlled. Moreover, θ is computed based on member dynamism which reduces the number of merge/split and rekeying operations. The subsequent describe each module of ASAKE protocol in detail.

Impact prediction: The total number of members about to arrive at a particular subgroup is defined as the impact (Prakash *et al*., 2016). The objective of the impact prediction mechanism is to predict the time of impact on a subgroup. This involves predicting the average dwell time of a member in a subgroup and number of members moving from one subgroup to other within a multicast group. Hou and Fang (2001) introduced the notion of Influence Curve in cellular networks. Let s(t) denote the subgroup-dwell-time probability density function and it is assumed that s(t) is known or can be computed by any Core. Let m_k^i denote a member k belonging to subgroup z_i. If the multicast group member enters the subgroup at time t, the probability that the member will leave the subgroup after time T is given by:

$$\Pr\left(m_k^i \in z_i \,|\, t < T\right) = \int_0^{T-t} s(\tau)d\tau = L(t, T)$$

Let $\alpha(i, j)|j \in N_i$ be the probability that the member moves to subgroup $Core_j$ from $Core_i$ where $\sum_{j \in Ni} \alpha_{i,j} = N_i$ is the set of the neighbouring subgroups to the subgroup i. For a totally random movement pattern in a homogenous multicast group, the members move to all possible directions or neighbouring subgroups with equal probabilities, $\alpha_{i,j} = 1/N_i$ for all $j \in |N_i|$. where $|N_i|$ denotes the cardinality of the set N_i. With L(t, T) and $a_{i,j}$, the impact of a leaving member can be defined as I(i, j, t, T) = $a_{i,j}$ L(t, T).

The above impact definition characterises the impact exerted on subgroup j at time T by a member leaving subgroup i which entered at time t. The more the impact exerted on a subgroup j at time T, more likely a rekeying has to be performed at time T in subgroup j. The total impact of all the leaving members from subgroup I to j is:

$$I_{i,j} = \omega \Sigma_{k \in S} \alpha_{i,j} L(t_k, T)$$

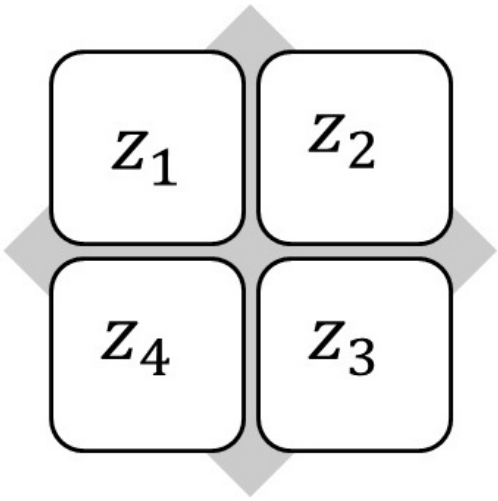

Fig. 4: Impact prediction

where, ϵ is a tuneable constant. The other mobility parameters like newly arriving membersand members arriving due to merge/split decision do constitute to the impact. ASAKE considers them in the Rekey Interval Computation mechanism described in the next sub section (Fig. 4).

Rekey interval computation: The rekey interval computation mechanism computes the rekey time T_{rekey} and rekey interval θ_i for a subgroup i. Fig. 5 shows a timeline depicting the possible rekey time instants (t_n) and the overhead associated with each rekey. The objective is to minimise the rekey overhead during the period (t_0, $t_0+\theta$) denoted by E[C]. t_0, t_1, …, t_n denote the predicted time of events such as member arrivals (λ), member leave due to expiry of service duration or member expel, predicted join impact from neighbouring subgroups and predicted leave impact due to members leaving the subgroup to another. If a rekey is performed for each event at]. t_0, t_1, …, t_n then the total overhead during the period of consideration is given by:

$$E[C] = E[C_0]+E[C_1]+...+E[C_n]$$

where: $E[C_n]_t$ is the sum of rekey overhead due to join, leave and transfer requests at the moment given by $E[C_n]_t = E[J_n]+E[L_n]+E[T_n]$. The objective is to minimise the total overhead E[C] by delaying the process of rekeying dynamically (within tolerable limits of members) and thereby performing a combined rekey for multiple membership events. This can significantly reduce the rekey overheads (Prakash *et al*., 2016).

It is assumed that each subgroup i sends $\{I_{i,j}, T, \lambda_i\}$ to its neighbours periodically to ensure updated information (where, $I_{i,j}$ is the total impact of all the leaving

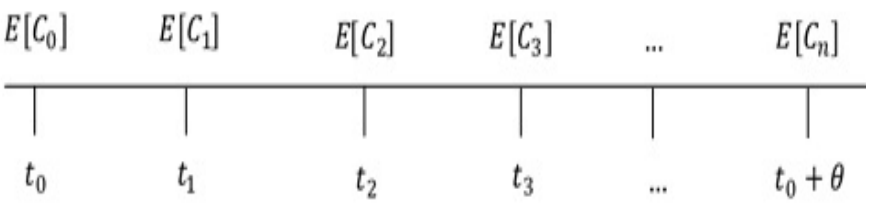

Fig. 5: Timeline-possible rekey time instants and overhead

members from subgroup i to j, T is the time of impact computed by the impact prediction mechanism and λ_i is the arrival interval in that subgroup). This information need not be sent explicitly, it can be piggybacked on multicast data packets that are being forwarded to the neighbouring subgroups/subgroups down the tree. This information is not only useful for key management but can also be used for multicast routing and QoS provisioning. It is assumed that the subgroups are time synchronised and each core can calculate the time synchronisation error. The rekey time computation algorithm is described below:

- $T_{join} = m_t^L = m_t^J = \infty$ where m_t^x denotes the first member request in the queue received at time t and x denotes either join request (J) or leave request (L). Also, T_{join} denotes the maximum impact occurring time on subgroup i by a neighbouring sub group j among the impact updates received during $(t_0, t_0+\theta)$ where t denotes the current time and t_0 denotes the last rekey time
- If there is an impact during $(t_0, t_0+\theta)$ then set T_{join} with the time of maximum impact from other neighbouring subgroups and its given by:

$$T_{Join} = T\left(\max_{j \in N_i} I_j\right) \Big| t_0 < T < t_0 + \theta$$

- If RAQ is not empty at t_0 then update m_t^L with the time of the first arrived leave request and m_t^J with the time of the first arrived join request
- $t_{rekey} = \max(p\theta, q\lambda) | t_{rekey} \leq \lambda$ where γ denotes the tolerable latency, p and q are tuneable constant set by the application
- If $t_{rekey} \leq \gamma$ then $T_{rekey} = t_0 + t_{rekey}$ Else $T_{rekey} = t_0 + \gamma$.
- Update $t_0 = T_{rekey}$ and $\theta = T_{rekey} - t_0$.

Therefore, the above algorithm sensibly delays the rekey process based on the dynamism and thereby reduces the number of rekeying by combining events.

Merge/split decision: Group merge-split algorithm adapts the architecture dynamically to reduce the number of subgroups in the multicast session with respect to member dynamism. The group merge-split decision algorithm shown in Fig. 6 is executed periodically for every θ_i time units in each subgroup i. θ_i is the rekey interval computed as described in the previous subsection.The periodicity θ_i enables group merge-split to happen with respect to member dynamism and hence will reduce the number of redundant group merge-split operations.

Let λ_i denote the arrival rate of the subgroup i and λ denote the average arrival rate of the neighbouring subgroups of subgroup i. Let cλ' be the threshold or acceptable dynamism in subgroup, where c, a control parameter constant. The merge/split decision algorithm works as shown in Fig. 6. When λ_i becomes greater than λ' the dynamic portion of the subgroup i is isolated or the subgroup i is split reducing the effect of 1-affects-N problem. The group split algorithm initialises the cryptographic keys for operation.

When the arrival rate of the subgroup i is stable and equal to λ for bθ_i periods, then the subgroup i is merged with its parent subgroup. This reduces the number of subgroups and hence the key translation overhead. b is a control parameter constant. It should be noted that the number of merge-split operations can be controlled by the control parameters c and b desirably.

Rekeying mechanisms for membership events: The membership events trigger rekeying and ASAKE rekeying strategy is shown in Fig. 7. The members send the Join, Leave and Transfer requests to the respective Core. Core accumulates all the member join requests for θ time period and then rekeys. The member leave requests to leave the multicast session are handled instantly to ensure forward secrecy. The member transfer request requires member join rekeying in the new subgroup and member leave rekeying in the current subgroup. To reduce the transfer latency and uninterrupted service in dynamic networks to

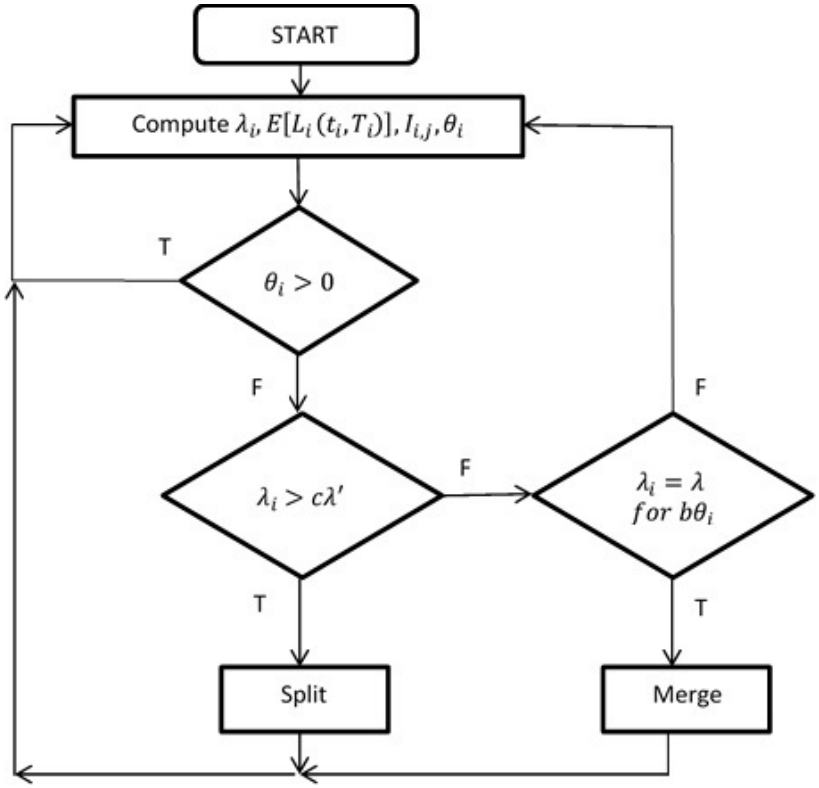

Fig. 6: Merge split decision algorithm

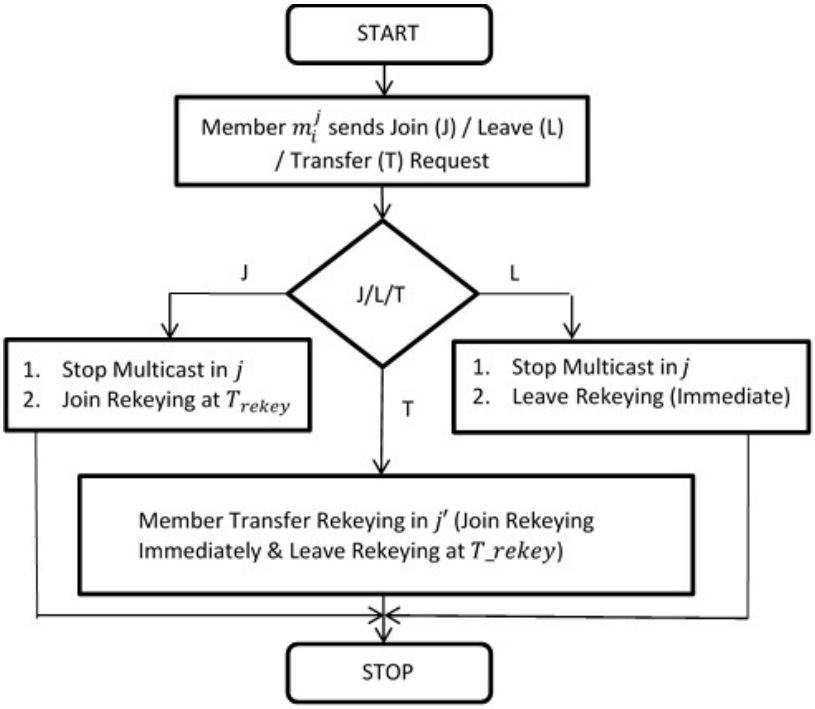

Fig. 7: ASAKE rekeying machanism

mobile members, member join rekeying arising due to member transfer is done immediately. But member leave rekeying due to member transfer in the current subgroup is done after θ or at T_{rekey}. Each membership event's rekeying mechanism of ASAKE with KEK_M and KEK is described below.

Initial group setup: The initial group setup is the process of setting up the group for the first time for a multicast session. Let $mm_j = \{m_1, \ldots, m_y\}$ denote the set of members wishing to join the multicast subgroup $z \in G | j = 1$ initially. z is under the control of the core denoted by Core. Initially, the multicast session starts with one group and subsequently many subgroups are created based on Group Merge-Split Decision algorithm.

Algorithm 1: Asake initial group setup:

$\forall m_i | (m_i \in mm_j \text{ and } i = 1, \ldots, y)$

$m_i \rightarrow \text{Core: } \{JOIN\}^{ucast}_{KU_{Core}}$

$\forall m_i | (m_i \in mm_j \text{ and } i = 1, \ldots, y)$

Authenticate (m_i^j)

$\text{Core: } KEK_M = \Gamma_{m_x} + \ldots \Gamma_{m_y}$

$\text{Core} \rightarrow z: \{KEK\}^{mcast}_{KEK_M}$

$\text{Core} \rightarrow z: \{Data\}^{mcast}_{KEK}$

The initial group setup is given in Algorithm 1. The equation A→B: $\{Message\}_K^{ucast}$ denotes an entity A sending B the Message encrypted with K and sent as a unicast message and this convention is used in describing the algorithms in the following subsections.

The member's sub-keys ($\Gamma_{(m, i)} | = 1 \ldots$) are added as shown in step 3 of Algorithm 1. The multicast service is started after establishing the encryption keys KEK_M and KEK which just requires sending the KEK encrypted with the KEK_M as a multicast message. ASAKE's initial group setup algorithm needs to send only one multicast message to the corresponding subgroup as given in step 4 of Algorithm 1.

Group merge: Assume z_a and z_b are subgroups while z_a is the parent of z_b and merge decision is taken.

Algorithm 2: Asake merge:

Algorithm 2 describes the procedure involved in merging the subgroup z_b with z_a. Let z_b be the subgroup given by:

$\text{MERGE}(z_\alpha, z_b)$

$z_b = \{m_1^b, \ldots m_y^b\}, V_b = \{v_1^b, \ldots v_y^b\}, \Gamma^b$

$= \{\Gamma_1^b, \ldots \Gamma_y^b\}, S_k^b = \{S_{k_1}^b, \ldots S_{k_y}^b\}$

$\text{Core}_{za} : \text{Authenticate}(\text{core}_{zb})$

$\text{Core}_{zb} \rightarrow \text{Core}_{za}:$

$\{KEK_M^b, \Gamma^b, \text{Core}_{zb}, \text{Core}_{z\alpha}, T\}^{ucast}_{SK_a}$

$\forall m_i^b \in z_b : KEK'_{z_a} = KEK_{za} + v_i^\alpha . S_{k_{m_i^b}}$

$\text{Core}_{z\alpha} \rightarrow m_i^b:$

$\left\{\Gamma_i^\alpha = \{v_i^\alpha . S_{k_{m_i^b}}\}, TEK_{z\alpha}, \text{Core}_{z\alpha}, m_i^b, T\right\}^{ucast}_{S_{k_{m_i^b}}}$

$z_b \{m_1^b, m_2^b, m_y^b\}$ where $\{m_i^b \in z_b | i = 1, \ldots y\}$. Corez_a which is the parent of z_b, authenticates Corez_a by executing the authentication algorithm (R). After successful authentication, Corez_b sends a unicast message containing the following:

- $KEK_M{}^b$, key encryption key used in the subgroup Corez_b before merging and $\Gamma b = \{\Gamma_1^b, \ldots \Gamma_y^b\}$ the set of individual members sub-keys.
- Identity of Corez_b and Corez_a
- Timestamp T

All the above encrypted with the secret key SK_a shared between Corez_a and Corez_b as described in step 2 of Algorithm 2. Then, z_a computes the new KEK including the members of subgroup z_b in the master key and distributes the new KEK to the new group members. From Algorithm 2 it can be observed that a Group Merge algorithm requires y unicast messages to be sent for each operation.

Group split: Assume a cluster Ci = $\{z_a, z_b\}$. When the decision to split is taken, the ASAKE Group Split is executed.

Algorithm 3

Asake group split:

$z_b = \{m_1^b, m_2^b, m_y^b\}, KEK_M^b$

$= \{\Gamma_1^b + \Gamma_2^b + \ldots \Gamma_y^b\}$

$\text{Corec}_i \rightarrow C_i : \{\text{CORE\_LEAVE}\}^{mcast}_{KEK_M^b}$

$\text{Corez}_b \rightarrow \text{Corec}_i : \{JOIN\}^{ucast}_{KU_{corec_i}}$

$\text{Corec}_i \text{Authenticate}(\text{Corez}_b)$

$\text{Corec}_i \rightarrow \text{Corez}_b:$

$\{KEK_M^b, \Gamma^b, \text{Corez}_b, \text{Corec}_i, T\}^{ucast}_{Sk_a}$

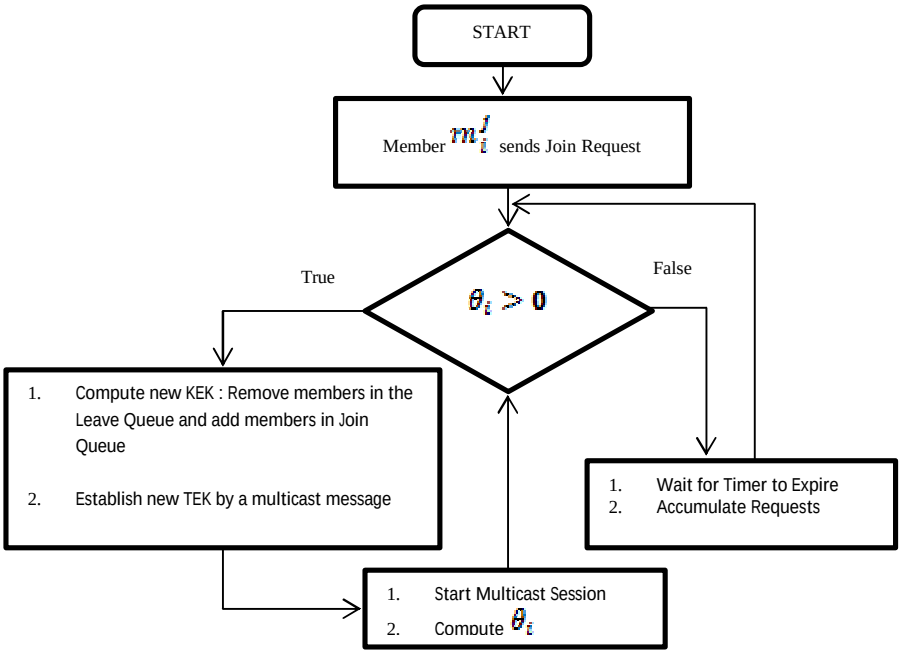

Fig. 8: ASAKE join rekeying

Algorithm 3 describes the procedure involved in splitting the subgroup z_b from C_i. Let z_b denote the subgroup to be split from the cluster given by $z_b = \{m_1^b, m_y^b\}$ where, $(m_i^b \in z_b | i = 1, \ldots, y)$ denotes members belonging to the subgroup z_b and the number of members denoted by y.

The splitting of the subgroup z_b from the cluster's core $Core_{Ci}$ starts by $Core_{Ci}$ sending a $CORE_{LEAVE}$ message to members of subgroup z_b. The $CORE_{LEAVE}$ message triggers the leader election algorithm and a $Core_{zb}$ is elected eventually. The newly elected $Core_{zb}$ joins the cluster by sending a join message followed by authenticating itself to $Core_{Ci}$. Using the secret key transferred securely during the authentication process, the $Core_{Ci}$ sends the KEK_M^b, $|^b$, identification information of cores (C_i, z_b) and timestamp T to $Core_{zb}$ as shown in step 5 of the above algorithm. Then, $Core_{Ci}$ adds $Core_{zb}$ to its current KEK and removes z_b's key encryption key, KEK_M^b from its current KEK as shown in step 6 of the Algorithm 3. Both the Cores then generate and distribute new TEK to their groups by sending a multicast message containing the new TEK encrypted by the corresponding subgroups current KEK as shown in step 7 and 8 of Algorithm 3. It can be observed from Algorithm 3 that it requires 2 multicast messages and one unicast message (if necessary).

Member join: The Member Join Rekeying mechanism is invoked whenever there are only join requests or leave requests (transfer leave) in the RAQ. The objective of the member join algorithm is to include the new member in the subgroup ensuring backward secrecy. Also, if there are any leave requests (transfer leave) in the RAQ, the member join algorithm eventually excludes the remaining members of the subgroup ensuring forward secrecy. The member join rekeying is predictive rekeying mechanism taking place every θ_j period and Fig. 8 shows the overall working of join rekeying mechanism while Algorithm 4 describes the algorithm.

Algorithm 4:
Asake member join

$$Core_j \rightarrow z_j : \{D\alpha t\alpha\}_{TEK}^{mcast}$$

$$m^j \rightarrow Core_i : \{JOIN\}_{KU_{corei}}^{ucast}$$

$$Authenticate(m^j)$$

$$Core_j KEK'_M = KEK_M +$$

$$(\Gamma_1 + \ldots + \Gamma_{join_m}) - (\Gamma_1 + \ldots + \Gamma left_m)$$

$$Core_j \rightarrow z_j : \{TEK'\}_{KEK'_M}^{mcast}$$

$$Core_j \rightarrow z_j : \{D\alpha t\alpha\}_{TEK'}^{mcast}$$

Let $(\Gamma_{jq} = \Gamma_1 + \ldots + \Gamma join_m)$ be the set of members whose join request and $(\Gamma_{lq} = \Gamma_1 + \ldots + \Gamma left_m)$ be the set of leave requests (transfer leave) accumulated during the period θ in RAQ. Using Multicrypt, Algorithm 4 revokes the users who have sent leave request and dynamically includes the

members who have sent join request. Step 4 of Algorithm 4 depicts it and then the multicast communication is stalled to establish the newly generated TEK' as shown in step 6 of algorithm 4. Hence, ASAKE member join rekeying algorithm requires sending just one multicast message for single member join or multiple member join event.

Member leave: Member leave rekeying is immediate rekeying and so each member's leave request is handled instantly. The Member Leave algorithm uses the property of Multicrypt that any member can be dynamically revoked with trivial computations. Step 3 of Algorithm 5 shows the dynamic user revocation where, m^j is the user and Γ_m^j is the corresponding sub-key that needs to be revoked. The multicast service is stopped before rekeying and resumed the process of rekeying. ASAKE's member leave algorithm needs to send only one multicast message to the corresponding subgroup as given in step 5 of Algorithm 5 to revoke a member from the multicast group.

Algorithm 5
Asake member leave:

$$\text{Core}_j \rightarrow z_j : \{D\alpha t\alpha\}_{TEK}^{mcast}$$
$$m^j \rightarrow \text{Core}_i : \{JOIN\}_{KU_{corei}}^{ucast}$$
$$\text{Core}_j KEK_M^{'} = KEK_M - \Gamma m^j$$
$$\text{Core}_j \rightarrow z_j : \{D\alpha t\alpha'\}_{TEK'_M}^{mcast}$$
$$\text{Core}_j \rightarrow z_j : \{D\alpha t\alpha\}_{TEK'}^{mcast}$$

Member transfer: When a member of a multicast group m_i^a belonging to a subgroup z_a under the control of the $Corez_a$, moves to a position which is in close proximity to a different core, say, $Corez_b$. The corresponding member can be transferred to z_b which would reduce a lot of unnecessary message transmissions over the network and end-to-end delay in the delivery of multicast service. In certain situations the member would have already moved into the area of neighbouring subgroup and needs transfer. The member transfer algorithm is described in Algorithm 6. Let m_i^a be the member belonging to the subgroup z_a which had moved/tends to move to subgroup z_b ($Corez_b$).

Algorithm 6
asake member transfer:

$$m_i^{\alpha} \rightarrow Corez_b, Corez_{\alpha} :$$
$$\{TRANSFER\}_{KUCore_{zb}}^{ucast}$$
$$Corez_{\alpha} : Authenticate(Corez_b)$$
$$Corez_{\alpha} : Addm_i^{\alpha} toRAQ(leavequeue)$$
$$Corez_{\alpha} \rightarrow Corez_b :$$
$$\left\{S_{K_{m_i}}, Core_{zb}, Core_{z\alpha}, T\right\}_{S_{k_b}}^{ucast}$$
$$Corez_b : KEK_M^{b'} = KEK_M^b + \Gamma_{m_i^b},$$
$$where \Gamma m_i^b = v_i . S_{k_{m_i}} \mid vi \in V^b$$
$$Core_{zb} \rightarrow m_i^{ab} : \left\{\Gamma_{m_i^b}, Core_{zb}, TEK^b, T\right\}_{Sk_{m_i}}^{ucast}$$

The member m_i^a sends a member transfer request to $Core_{za}$ and $Core_{zb}$. $Core_{za}$ then authenticates $Core_{zb}$ if there were no previous transaction between them in the recent past. $Core_{za}$ adds m_i^a to the leave queue of RAQ to be removed from the subgroup in next rekeying. Also, ? sends a unicast message containing the secret key $S_{k_{m_i}}$ computed during authentication of moved member, the identity of ?, the identity of $core_{zb}$ and the time stamp (T) all encrypted with S_{k_b} the secret key computed during authentication of $core_{zb}$ as shown in line 4 of the transfer algorithm. The new key encryption key ($KEK_M^{b'}$) is then computed by $core_{zb}$ as given in line 5 of the transfer algorithm.

The $core_{zb}$ then sends a unicast message containing $\{\Gamma m_i^b, core_{zb}, TEK^b, T\}$ where $\Gamma_{m_i^b}$ is the new private key of m_i^{ab}, TEK^b is the traffic encryption key of subgroup z_b as indicated in line 6 of transfer algorithm. The member will belong to z_a and z_b until the next leave rekeying at z_a and eventually will be deleted from the subgroup or after

$n\theta_{zz}$ time units whichever is earlier. n is the tuneable constant which can be appropriately set based on the security concern. The algorithm 6 shows that each member transfer requires 2 unicast messages to be sent. It should be noted that the unicast message in step 4 will not be required all the time.

RESULTS AND DISCUSSION

Performance analysis: This section compares and analyses the overhead incurred by various standard key establishment schemes. The performance of the static and adaptive protocols are analysed through simulation.

Complexity analysis: Rekeying induces computational and communicational overheads. The communicational overheads are due to the distribution of rekey packets to preserve forward and backward secrecy during

Table 1: Overhead Complexity

	Centralised			Decentralised			Key Agreement		
	GKMP	LKH	Secure Locks	IOLUS	Kronos	ASAKE	GDH	DH-LKH	Fiat
Join multi	2	Log_2 (n)-1	0	2	2	1	n	$\log_2 n$	n
Rekey cast									
Unicast	2	$\log_2 n+1$	2	1	1	0	n-1	0	n
Multi-Join Rekey	4y	$(2\log_2 n+1)y$	2y	3y	3y	1	(2n+1)y	$(\log_2 n)y$	ny
Leave Rekey	2y	$2\log_2 n$	0	(n/z)	(n/z)	1	n+(n-1)	$\log_2 n$	n
Multi-leave rekey	(2n)y	$(2 \log_2 n)$ y	-	(n/z) y	(n/z) y	1	(2n+1) y	$(\log_2 n)$ y	ny
Storage-server	n+2	2n-1	2n	(n/z)	(n/z)	(n/z)	-	-	-
Storage-member	3	$\log_2$ (n)+1	2	2	2	2	-	-	-
Key arrangement	Pairwise	Tree	Broadcast	Hierarchical	Hierarchical	Hierarchical	Ring	Tree	Broadcast

y-number of joining/leaving members, n-degree of the tree, z-number of subgroups

membership change. The computational overheads are mainly due to encryptions of the rekey packets. The rekeying mechanisms can be classified into three major classification based on the key management control as centralised, decentralised and key agreement. Again, they can be further sub classified based on the key arrangement as pairwise, hierarchy, broadcast and ring. GKMP is the standard pairwise centralised rekeying mechanism, LKH is a better performing tree based centralised rekeying mechanism, Secure Locks is a broadcast centralised rekeying mechanism, IOLUS is hierarchical decentralised rekeying mechanism (Setia *et al*., 2000) is hierarchical decentralised periodic rekeying mechanism, ASAKE is hierarchical decentralised rekeying mechanism, GDH is hierarchical key agreement, DH-LKH is tree based key agreement and Fiat is broadcast key agreement rekeying mechanisms. The selected protocols are the best solutions in their main and sub category. It can be observed from Table 1 that ASAKE performs better than all other considered protocols.

Simulation: The main objective of the simulation is to analyse the bandwidth saved by ASAKE, adaptive key management protocol and the effect of control parameters of adaptive key management protocols.The simulation is performed with the following performance parameters.

Packets sent: The number of application packets sent in a multicast session by the application or traffic generator.

Rekey packets: The cumulative sum of rekey packets sent due to the membership view change during a multicast session.

Normalised Rekey Load (NRL): Normalised rekey load is the ratio of the total number of application packets received per node to the total number of rekey packets sent during a multicast session:

$$NRL = \frac{\text{Recieived packets}}{\text{Rekey Packets Sent} \times \text{Nodes}}$$

Rekey Load Saved (RLS): Rekey load saved is the ratio between the number of rekey packets sent with z subgroups of scheme A to the number of rekey packets sent with z subgroups of scheme B. The ratio is multiplied by hundred to indicate in percentage. The following equation gives the rekey load saved by decentralised architecture over centralised in percentage:

$$RLS = \left(1 - \frac{\text{Rekey Packets Sent}_{decent}}{\text{Rekey Packets Sent}_{cent}}\right) \times 100$$

Bandwidth utilised: Bandwidth utilised is measured by the following equation:

$$B = zSE[Prekey] + zR$$

Where:

B = Is the bandwidth utilised
S = Is the size of the data packet,
$E[P_{rekey}]$ = Is the expected number of rekey packets sent,
z = Is the number of subgroups (z = 1 for centralised schemes) and
R = Is the data packet rate (Chan and Chan 2002).

Bandwidth saved: Bandwidth saved is the amount of bandwidth saved. The following equation gives the bandwidth saved by decentralised scheme over centralised scheme:

$$BS = \left(1 - \frac{B_{decent}}{B_{cent}}\right) \times 100$$

The simulation is performed using NS 2.34 on Linux operating system running on an Intel i5 Pentium 2.44 GHz processor with 4GB RAM. The simulation parameters shown in Table 2 are used.

The impact of number of nodes on normalised rekey load, bandwidth utilised and number of application

Table 2: Simulation Pparameters

Parameters	Value
μ(req./s)	1/500
R(kbit/s)	5 kb sec^{-1}
β	1.33×10^{-3}
Simulation area (m)	1000×1000
Simulation time (s)	900
Max Number of Nodes	100
Max Number of Groups	Variable
Traffic Type	CBR(UDP)
Payload Size (B)	64

packets sent was computed and the performance of ASAKE is then compared with IOLUS, a decentralized standard key management protocol often used as a performance benchmark and GKMP, a centralised standard key management protocol. The best protocol from each category mentioned in table 1 was selected for comparison through simulation. Key agreement based protocols (Amir *et al*., 2004) are functionally same as that of centralised protocol and hence GKMP is selected to represent both categories.

Impact of nodes: The impact of number of nodes on the performance of the protocols under consideration is observed. The objective is to evaluate the scalability of the protocols by varying the number of nodes. A protocol is said to be scalable when its performance does not degrade with the increase in the number of nodes or members in a multicast group. The simulation is carried out by varying the number of nodes while keeping the arrival rate, member duration, message interval and simulation duration constant. Figure 9 shows the normalised rekey load of GKMP, IOLUS and ASAKE protocols. It can be observed from Fig. 9 that the normalised rekey load of ASAKE is better than IOLUS. The Normalised Rekey Load (NRL) is the number of rekey packets per received multicast packet. A reduction in the value of NRL indicates a reduction in the value of number of rekey packets and increase in the number of received multicast packet for given fixed data rate of multicast transmission. The efficient member transfer algorithm and predictive rekeying strategy used in ASAKE has decreased the ASAKE-NRL when compared to IOLUS and GKMP.

Reduced frequency of rekeying results in minimum interruption in the multicast service as the service will be stalled during rekeying and reduced effect of out-of-sync problem due to network latency or partitions. The reduced frequency of rekeying also helps to save bandwidth occupied by rekey packets. Figure 9 also shows the bandwidth saved by ASAKE compared to other protocols under consideration. It should be noted that the bandwidth is also saved when the number of key translation or decryption/encryption is reduced.

The number of key translation per multicast packet can be reduced if the number of groups is reduced. ASAKE's adaptive architecture strategy adapts the number of groups dynamically depending on the number of members balancing the 1-affects-N and key translation problem. The combined result of rekey load saved and bandwidth saved can be observed in Fig. 9. It also shows 1-affects-N and key translation problem is balanced to produce efficient utilisation of the bandwidth.

Impact of control parameters: The impact of control parameters on the performance of the protocols under consideration is observed. The objective is to evaluate the controllability of ASAKE by varying the control parameters. The simulation is carried out by varying control parameters p and q while keeping the number of nodes, member duration, message interval and simulation duration constant. Figure 10 shows the normalized rekey load andjoin latency for fixed number of nodes while varying the control parameter p and setting q = 1. While Fig.11 shows the normalized rekey load andjoin latency for fixed number of nodes while varying the control parameter q and setting p = 1. Figure 10 shows the normalized rekey load variations with the increase the control parameter p. It can be observed that a small reduction in the normalised rekey load can lead to significant reduction in the number of rekey packets which can be observed from Fig. 10. The reduction in rekey packets is due to increase in value of p which results in increased period of interval between two rekeys. Hence the frequency of rekeying can be adjusted desirably with p.

The simulation is performed with member transfer interval fixed to one simulation seconds. Therefore, if the period of interval between two rekeys increases then the frequency of rekeying decreases. Hence the rekey packets and normalised rekey load also decreases. But it can be observed from Fig. 10 that the Join latency increases linearly with the increase of p and the periodic rekey interval. Therefore, a longer rekey periodic interval causes lesser rekeying but increased Join latency. In other words, a higher value of p will give weightage to existing members of the multicast group while a lower value of p will give weightage to newly joining members of the multicast group. Figure 11 shows variation of control parameter q from which it can be observed that the figures exhibit similar characteristics of variation of control parameter p. Thus, it can be inferred that by adjusting the value of p and q, the weightage given to service quality of the existing members to that of newly joining members can be obtained desirably.

Security analysis: The proposed ASAKE protocol provides Authentication, Key Confirmation and Forward Secrecy. The following sub-sections analyse the security of ASAKE.

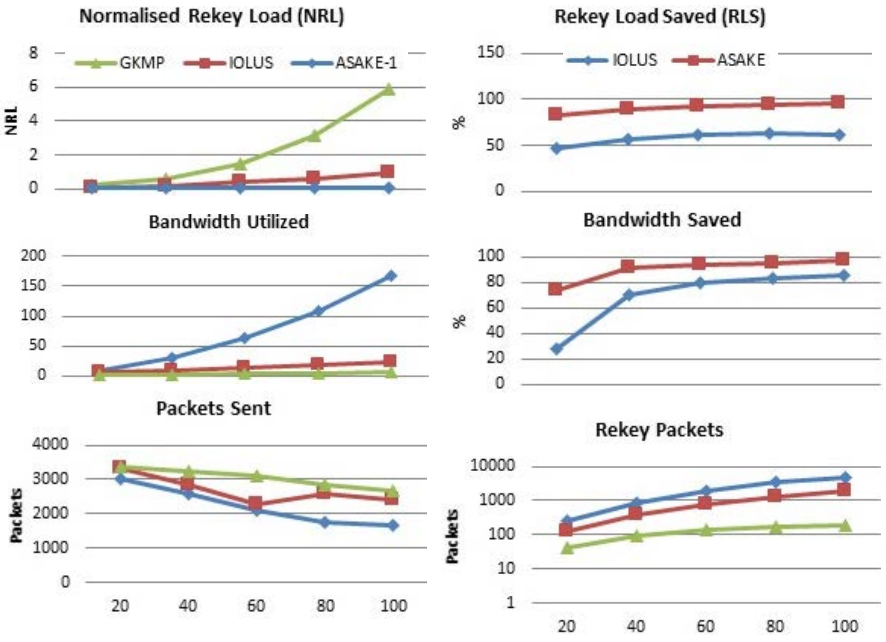

Fig. 9: ASAKE performance variable nodes

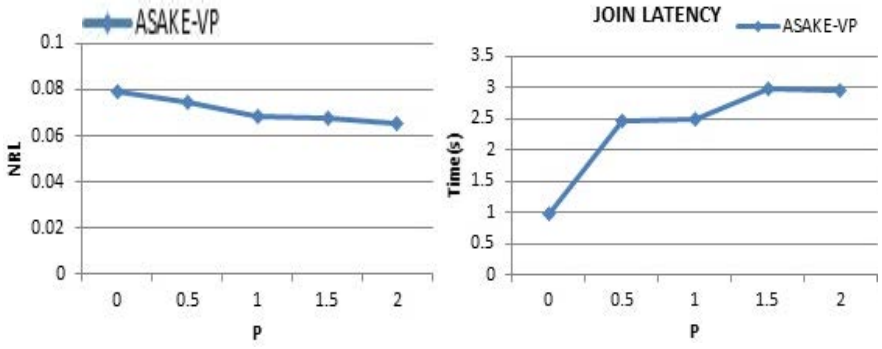

Fig. 10: Control parameter responses (VP)

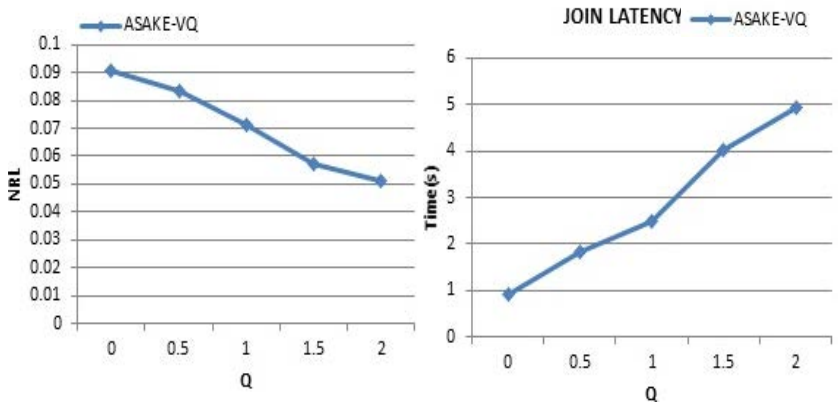

Fig. 11: Control parameter responses (VP)

Authentication, key confirmation and forward secrecy: The authentication is integrated in to the protocol itself by the use of Multicrypt. But any authentication mechanism like (Nguyen and Roscoe 2011) can be used instead. Consider the following theorem originally stated by Boyd *et al*. (2006).

Theorem 1: Assuming the Computational Diffie-Hellman (CDH) assumption is satisfied in Q, the protocol is a secure key agreement protocol providing key confirmation and forward secrecy when H_1 and H_0 are modeled as random oracles and if the underlying message authentication scheme and encryption scheme are secure in the sense of existential unforgeability under adaptive chosen-message attack and indistinguishable under chosen-plaintext attack, respectively.

Proof: The underlying encryption scheme was proved in the sense of existential unforgeability under adaptive chosen-message attack and indistinguishable under chosen-plaintext attack respectively. The potential candidate function for random oracle was identified to be MD5 which is a proven secure message authentication scheme. Considering the above proven facts the above theorem holds for ASAKE rekeying as well. Moreover, the authentication algorithm is a modified authentication procedure and an exhaustive security analysis can be found in Boyd *et al*. (2006).

Forward and backward secrecy: Forward secrecy requires that users who had left the group should not have access to any future keys. This ensures that a member cannot decrypt data after it leaves the group. When a member leaves the group, the encryption key is updated as $K = K-\Gamma_i$, where the sub-key (Γ_i) of the leaving member m_i is removed from the encryption key. Backward secrecy requires that a new user that joins the session should not have access to any previous transactions.

Theorem 2: A message m encrypted with encryption key K can be decrypted only by using either the encryption key or only by any of the sub-keys that are included in the encryption key K.

Proof: The encryption key is given by:

$$K = v_1\Gamma_1 + \ldots + v_i\Gamma_i + v_i + 1 + \ldots + v_{p-1}\Gamma_{p-1}$$

Assume member m_i leaves the group, $v_i\Gamma_i$ which is the sub-key of the leaving member m_i, will be removed from the encryption key. Then the updated encryption key is given by:

$$K = v_1\Gamma_1 + \ldots + v_{i-1}\Gamma_i + v_{i+1} + \ldots + v_{p-1}\Gamma_{p-1}$$

Let the message be m and the cipher text be c. Let us assume that the recently revoked member m_i tries to decrypt the message violating the forward secrecy. Then the decryption process is given by:

$$D\Gamma_i(c) = \frac{c.v_i^T}{\Gamma_i} = 0 \text{or } \perp$$

Since, $\{(\Gamma_1 + \ldots + \Gamma_{i-1} + \ldots + \Gamma_{p-1}).w + vj.g^{sy}.r\}v_i^T$ By the Principle of Orthogonality, $v_i.v_j^T = 0, \text{if} i \neq j$ where i, j can take values between 1, …, p-1. In other words, $vi.v_j^T = 1$ if i = j. Hence, the revoked member cannot decrypt the message correctly.

CONCLUSION

This paperproposed an Adaptive Secure Authenticated Key Establishment (ASAKE) protocol. ASAKE is a lightweight protocolthat provides secure multicast communications in a dynamic network environments. ASAKE is based on Multicrypt Cryptosystem with new strategies of dynamic adaptation of architecture. ASAKE reduces significantly the reaction of key establishment to mobility. ASAKE reduces the effect of state-full member expectation, out-of-sync problem, overhead per rekey and the coupling between member dynamics and rekeying by using Multicrypt. ASAKE reduces the frequency of rekeying by using predictive rekeying mechanism which rekeys based on the rate of change of membership change. ASAKE's proposed adaptive architecture keeps the trade-off between 1-affects-n problem and key translation overhead in balance. ASAKE is designed to be tuneable to application requirements andfrom the simulation results obtained it can be observed that ASAKE has very less rekey overhead, very less service latency, bandwidth and work per data and control packets.

REFERENCES

Amir, Y., Y. Kim, C. Nita-Rotary and G. Tsudik, 2004. On the performance of group key agreement protocols. Trans. Inform. Syst. Secur., 7: 457-488.

Arockiasamy, J.P., L.E. Benjamin and R.U. Vaidyanathan, 2012. Performance evaluation of multicrypt encryption mechanism. Am. J. Appl. Sci., 9: 1849-1861.

Boyd, C., K.K.R. Choo and A. Mathuria, 2006. An Extension to Bellare and Rogaway 1993 Model: Resetting Compromised Long-Term Keys. In: Information Security and Privacy. Batten, L.M. and S.N. Reihaneh (Eds.). Springer Berlin Heidelberg, Berlin, Germany, ISBN: 978-3-540-35458-1, pp: 371-382.

Canetti, R., J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas, 1999. Multicast security: A taxonomy and some efficient constructions. Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies, March 21-25, IEEE Computer Society, New York, pp: 708-716.

Chan, K.C. and S.H. Chan, 2002. Distributed servers approach for large-scale secure multicast. IEEE. J. Sel. Areas Commun., 20: 1500-1510.

Daghighi, B., M.L.M. Kiah, S. Shamshirband and M.H.U. Rehman, 2015. Toward secure group communication in wireless mobile environments: Issues, solutions and challenges. J. Network Comput. Appl., 50: 1-14.

Hardjono, T. and G. Tsudik, 2000. IP Multicast Security: Issues and Directions. In: Annales Des Telecommunications. Hardjono, T. and G. Tsudik (Eds.). Springer-Verlag, Berlin, Germany, pp: 324-340.

Harkins, R., E. Weber and A. Westmeyer, 2005. Encryption schemes using finite frames and hadamard arrays. Exp. Math., 14: 423-433.

Hegland, A.M., E. Winjum, S.F. Mjolsnes, C. Rong, O. Kure and P. Spilling, 2006. A survey of key management in ad hoc networks. IEEE Commun. Surv. Tutorials, 8: 48-66.

Hou, J. and Y. Fang, 2001. Mobility-based call admission control schemes for wireless mobile networks. Wirel. Commun. Mobile Comp., 1: 269-282.

Mehdizadeh, A., F. Hashim and M. Othman, 2014. Lightweight decentralized multicast-unicast key management method in wireless IPv6 networks. J. Network Comput. Appl., 42: 59-69.

Nguyen, L.H. and A.W. Roscoe, 2011. Authentication protocols based on low-bandwidth unspoofable channels: A comparative survey. J. Comput. Secur., 19: 139-201.

Prakash, A.J. and V.R. Uthariaraj, 2009. Multicrypt: A provably secure encryption scheme for multicast communication. Proceedings of the First International Conference on Networks and Communications NETCOM'09, December 27-29, 2009, IEEE, Chennai, India, ISBN: 978-1-4244-5364-1, pp: 246-253.

Prakash, A.J., V.R. Uthariaraj and B.L. Elizabeth, 2016. Efficient key management protocol with predictive rekeying for dynamic networks. Proceedings of the 2016 2nd International Conference on Green High Performance Computing (ICGHPC), February 26-27, 2016, IEEE, Nagercoil, India, ISBN: 978-1-4673-6614-4, pp: 1-6.

Rafaeli, S. and D. Hutchison, 2003. A survey of key management for secure group communication. ACM Comput. Surv., 35: 309-329.

Setia, S., S. Koussih and S. Jajodia, 2000. Kronos: A scalable group re-keying approach for secure multicast. Proceedings of the IEEE Symposium on Security and Privacy, May 14-17, Berkeley, CA., USA., pp: 215-228.

Sterner, M., G. Tsudik and M. Waidner, 1996. Diffie-Hellman key distribution extended to groups. Proceedings of the Third ACM Conference on Computer and Communication Security, March 14-15, 1996, ACM Press, New Delhi, India,-pp: 31.

Wong, C.K., M. Gouda and S.S. Lam, 2000. Secure group communications using key graphs. IEEE/ACM Trans. Networking, 8: 16-30.