Distributed Server Networks for Secure Multicast

Kin-Ching Chan S.-H. Gary Chan
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon
Hong Kong

Abstract— **Multicast is an efficient technique to deliver data to a large group of users. For some applications offering multicast security is an important issue. In such a system, a new member should not be able to decrypt the multicast data sent before its joining and a former member should not be able to decrypt the mulitcast data sent after its leaving. Traditional approaches generally focus on reducing the re-key messages for a single server. However, these approaches still lead to large exchange overhead when the group is large. In this paper, we consider a distributed server network in which the user pool is split into multiple groups and served by multiple servers. Given the user traffic, there is a trade-off between the amount of re-key messaging and the total data bandwidth needed. We present a simple model for the system and study how the total bandwidth (including the re-key messaging and data traffic) can be minimized by optimizing the number of servers in the network. As the underlying user traffic is dynamic, a server should be able to split and merge user groups to minimize its total bandwidth. We propose a scheme for such a purpose. We show that distributed server network is able to substantively reduce the total bandwidth required in the system as compared to the traditional scheme.**

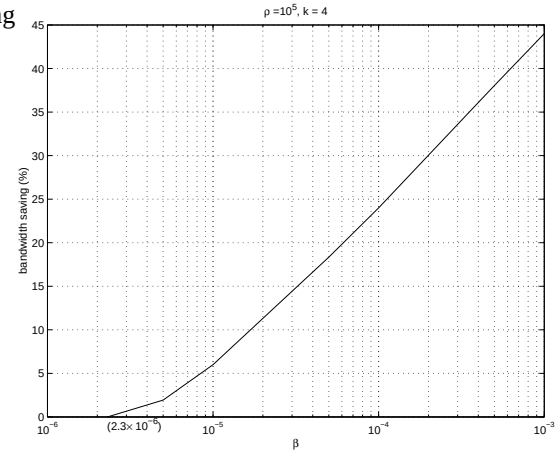

Fig. 1. An example of multicast system.

I. INTRODUCTION

Multicast is an efficient technique for delivering data to a large group of users in multimedia applications such as the Internet stock quote, Internet radio, audio/music delivery, video surveillance, etc [1]. Many of these applications requires data security. The current multicast protocols , however, do not offer any security features in terms of confidentiality, authenticity and integrity. In this paper, we would mainly study the data confidentiality issue in the multicast environment (i.e., unauthorized users should not be able to access the multicast data). In such a system, a new member should not be able to decrypt those multicast data sent *before* its joining (i.e., the so-called "backward secrecy") and a former member should not be able to decrypt those multicast data sent *after* its leaving or evicting (i.e., the so-called "forward secrecy") [2].

Traditional data security is generally based on PKI technology and applied in the unicast environment. Such a point-to-point approach is not applicable in the multicast environment with a large number of users, and when the group is highly dynamic, i.e., the group members join and leave frequently and at random times. Therefore, whenever there is a membership change in a group, the data has to be re-encrypted with a different key and the corresponding decryption key has to be made known to all the members in the group. If not managed properly, these "re-key messages" which inform the key change would consume a large amount of network bandwidth and processing overheads. An efficient solution to address this issue of key management has been proposed independently by Wong *et al*. and Wallner *et al*. [3], [4] Both schemes introduce a hierarchical key tree structure in which the group members are arranged as a logical key tree. Each group member is at the leaf of the tree and belongs to more than one multicast subgroups. Using this approach, the number of re-key messages for each change of membership (in the form of "join" and "leave") is shown to be only $O(\log N)$, where N is the number of concurrent users in the system, i.e., the group size.

In each server of a secure multicast system, there is generally a data manager and a control manager. The data manager encrypts and transmits data while the control manager is responsible for key management such as generating, storing and distributing keys. One issue of a single server system serving the whole population is that its complexity increases as the number of user increases, mainly due to the large number of re-key messages and the size of the key database (there are generally many keys involved in a secure multicast group). Therefore, in order to reduce the complexity and manageability of the system for a large multicast group, it may be beneficial to split the group into a number of smaller groups and serve them independently, thus forming a distributed server network. We show in Fig. 1 such a system, in which the servers distributed in the network serve their respective pools of users. Note that these servers are not necessarily geographically distributed — they may be logical servers. Therefore, a distributed server networks consists of one or more physical servers, each of which may contain one or more logical servers. Data is multicast to the users from its respective data manager, while the control manager notifies its users of the decryption keys.

Note that in such a server network, the total traffic for the multicast data is proportional to the number of servers. On the other hand, as the number of servers increases, the overhead in re-key messaging decreases (due to a decrease in the number of users served by each server). Therefore, there is a trade-off between re-key messaging and data traffic and a corresponding optimal number of servers such that the total bandwidth requirement of the servers is minimized given a certain user traffic. Clearly, when the data rate is high and the users are less dynamic (as in some video applications), splitting the pool of users into many groups may not be beneficial; on the other hand, if the data rate is low and the user pool is large and highly dynamic (as in stock quote applications), the pool of users is more likely to be split into many groups.

For a multicast application, traffic in a pool of users may not be consistently stationary. An example is an internet stock quote system. In general, user traffic is higher at the start of the day than the at end of the day. It is therefore important for the servers to split and merge their user groups dynamically so that the total bandwidth is minimized. The

This work is supported, in part, by the Areas of Excellence (AoE) on Information Technology funded by the University Grant Council in Hong Kong (AoE 98/99.EG01), and by the Hong Kong Telecom Institute of Information technology (97/98.EG01) in the HKUST.

0-7803-7206-9/01/$17.00 © 2001 IEEE

distributed server network is thus hierarchical in nature: the total pool of users is split and served by multiple independent physical servers where each server, depending on its local traffic, may further split and merge its user groups dynamically into multiple logical servers to minimize its own traffic. In this paper, we propose an efficient scheme for such splitting and merging.

There are three contributions of this paper: i) we present a simple model and analysis of the distributed server network for secure multicasting, which is supported by our simulation result; ii) we determine the optimal number of logical servers in order to minimize the total bandwidth in a system; and iii) we propose a dynamic split-and-merge scheme to reduce bandwidth requirement as the underlying user traffic is dynamic. Our result shows that for some applications such as a stock quote system, with low bit rate and a fairly large group of concurrent users (e.g., 100,000), our system can save a substantial amount of network bandwidth (up to 45% in some examples) as compared with a single server system.

Much of the previous work on secure multicast focuses on the key tree scheme. These works include reducing the number of re-key messages and the number of keys stored in the server [5], [6], [7]. All these works address mainly reducing re-key messages and have not considered a distributed server network nor the trade-off between multicast data and re-key messages. They mainly focuses on a specific number of users in the system and has no analysis on the re-keying cost when the users join and leave dynamically.

This paper is organized as follows. In Sect. II, we first review key the tree scheme and analyze the server network model using this scheme. In Sect. III, we show some illustrative numerical examples and results. In Sect. IV, we discuss the scheme on dynamically merge and split a multicast group. We summarize and conclude our research in Sect. V.

II. Scheme Description and Analysis

In [3] and [4], a hierarchical key tree approach is proposed to facilitate the distribution of re-key messages. They show that whenever a member joins or leaves the system the number of re-key messages is $O(\log N)$, where N is the number of concurrent users. In this section, we first review the scheme in Sect. II-A. and then present the analysis of the scheme (in terms of the average number of re-key messages that are generated when the users joins and leaves dynamically) in Sect. II-B.

A. Scheme Description

Hierarchical key tree is a logical tree structure of each multicast group stored in the control manager. In the tree, group members are arranged at the leaves and the internal nodes store keys (see Fig. 2 for a k-ary tree with depth d). There are three types of keys. The first one is a group key, K_1, used to encrypt/decrypt multicast data; the second one is a subgroup key (such as K_{d-1} and K_d) used to encrypt/decrypt other keys instead of actual data and the last one is an individual key, I. Each member holds the keys along the path from its leaf to the root. Therefore for the case of member u, u holds K_1,... K_{d-1}, K_d. Each subtree in the entire key tree is a subgroup and each member is assigned to more than one subgroup. For example, member u belongs to group G_d, G_{d-1},... , G_1.

Whenever there is a membership change, apart from the group key, all keys held by the new or former member have to be changed in a bottom-up manner. For example, if u leaves the group, first of all, we have to change K_d to a new subgroup key, say K'_d, and send it to all the members who shared K_d with u (i.e., u's sibling in the tree). Since K_d is known by u, the control manager has to encrypt K'_d by each members' individual key and send it to them by unicast. After sending K'_d, the process can be propagated one level up. Now, K_{d-1} has to be changed. Since K_d is changed to K'_d which is unknown for

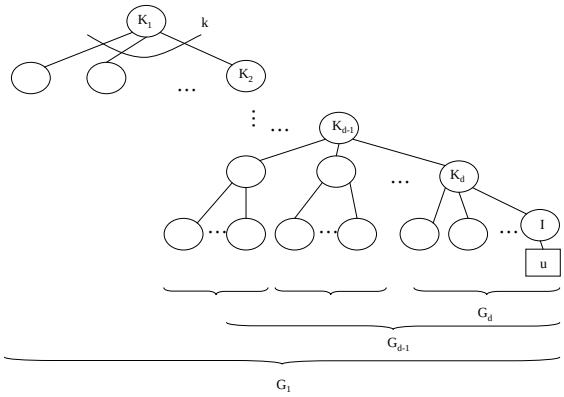

Fig. 2. A k-ary key tree.

u, the control manager can encrypt the new K_{d-1} with all subgroup keys including K'_d and send it to all subgroups in $d-1$ level. We repeat the same process upwards one level at a time until it reaches the root when K_1 is changed. Then all keys, including the group key, held by u are changed.

If u is a new member joining the group, in order to guarantee backward secrecy, all the keys from K_d to K_1 have to be changed. Since u knows nothing about the keys in the group, when the control manager changes K_d to the new key K'_d, then K'_d can be encrypted by K_d and multicast to u's sibling and unicast to u. Similarly, this process can be propagated upwards one level at a time, with the control manager multicasting the new keys to the subgroups under the key and unicasting the key to u.

If we assume that the key tree is a k-ary full tree, after each membership change, the number of re-key messages per leave and join are proportional to the depth of the key tree, $\log_k N$ where N is the group size. For each leave, each component of the key at each level has to be sent k times (one for each branch). For each join, each component of the key at each level has to be sent twice (one for multicasting to the old members while the other one for unicasting to the new member). Therefore, the number of re-key messages per leave and join are $k \log_k N$ and $2 \log_k N$, respectively.

B. Analysis

In this section, we analyze the system for the case in which users in a multicast group arrive according to a certain stochastic process with (target) rate λ (req/s). Each user stays in the system with with mean duration of $1/\mu$ (s). Define ρ as the average number of concurrent users in the system given by λ/μ. Let R bits/s be the data rate for a stream.

We consider that the pool of users is equally likely to access the m logical server, and hence the average number of concurrent users in a server is $\rho/m = \lambda/(m\mu)$. Denote S as packet size of a re-key message and $E[C_{\lambda/m,\mu}]$ as the cost of each server given by the expected number of re-key messages per second. Denote T bits/s as the total bandwidth used in the network, which is the sum of the re-key message data and multicast data. Clearly, T is given by,

$$T = mSE[C_{\frac{\lambda}{m},\mu}] + mR. \quad (1)$$

We are interested in minimizing T by adjusting m. To achieve this, we have done an analysis on the system.

To analyze our system, we consider that the requests arrive according to a Poisson process and the holding time is exponentially distributed. The system can therefore be modeled by a Markov process. Let $Q = \{0, 1, 2, ...\}$ be the system states indicating the number of

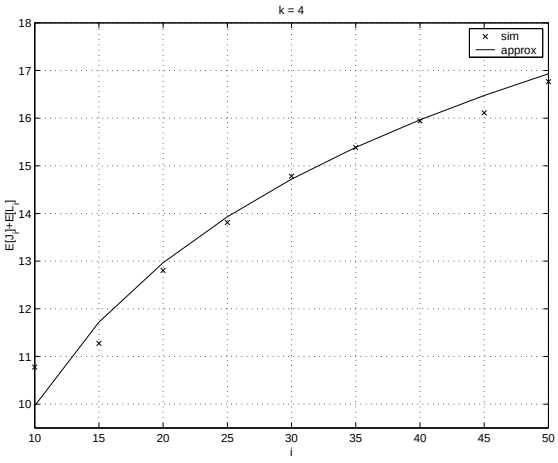

Fig. 3. $E[J_i] + E[L_i]$ vs i ($k = 4$).

concurrent users in a server. Let π_i be the steady state probability for the server in state i. It is well-known that $\pi_i = ((\frac{\rho}{m})^i/i!)e^{-\frac{\rho}{m}}$.

Note that a state change corresponds to a membership change which incurs some re-key exchange overheads (costs) in bits. Let J_i and L_i be the costs for a user joining and leaving the server in state i, respectively. Note that J_i and L_i are random variables depending on where the user join or leave the tree and are independent of ρ. Let $E[J_i]$ and $E[L_i]$ be the expected value of J_i and L_i, respectively. We show in Table I the nomenclature used in this paper.

By the long-run properties of Markov chain, $E[C_{\lambda/m,\mu}]$ can then be expressed by $E[C_{\frac{\lambda}{m},\mu}] = \frac{\lambda}{m}\sum_{i=0}^{\infty}\pi_i(E[J_i] + E[L_i])$, i.e.,

$$\frac{E[C_{\frac{\lambda}{m},\mu}]}{\mu} = \frac{\rho}{m}\sum_{i=0}^{\infty}\pi_i(E[J_i] + E[L_i]) \quad (2)$$

$$\triangleq f(m). \quad (3)$$

Therefore, T in Eq. (1) can be rewritten as $T = mS\mu f(m) + mR$, or, equivalently,

$$\frac{T}{R} \triangleq \sigma(m) \quad (4)$$

$$= m\beta f(m) + m \quad (5)$$

where β is a dimensionless parameters defined as $\beta \triangleq S\mu/R$. Equation (5) says that the total network bandwidth is known once $E[J_i]$ and $E[L_i]$ are obtained.

The closed-form expressions for $E[L_i]$ and $E[J_i]$ are intractable. Therefore, we consider that the key tree can be approximated to a full tree at any time. By considering the interesting case where ρ/m is large, we therefore have $E[L_i] = 2\log_k(i+1) \approx 2\log_k i$ and $E[J_i] = k\log_k(i-1) \approx k\log_k i$, where k is branching factor of the key tree. We show in Fig. 3 $E[J_i] + E[L_i]$ versus i for $k = 4$. The discrete points represent simulation results while the solid line is the analytical result. Clearly, simulation matches well with our analysis, showing that our approximation is valid. This is true even when the number of users are not large ($i \geq 15$).

Given ρ, $\sigma(m)$ (and hence T) in general first decreases and then increases as m increases. The expression of $f(m)$ is still quite complex and does not allow us to derive a closed form for m^*. Hence, we further make some approximations as follows. Observing that the

TABLE I
NOMENCLATURE USED IN THE PAPER.

Symbol	Definition
ρ	Total average number of concurrent users. $\triangleq \lambda/\mu$ (req./s)
λ	Arrival rate for the users (req./s)
μ	Average service rate
R	Date rate (bits/s)
S	Packet size of a re-key message (assumed constant) (bits)
β	$\triangleq S\mu/R$
m	Number of multicast groups in the system
$J_i, E[J_i]$	Random variable for the number of re-key messages and its expected value, respectively, for a new member joining the system with i concurrent users (msgs)
$L_i, E[L_i]$	Random variable for the number of re-key messages and its expected value, respectively, for a member leaving the system with i concurrent users (msgs)
$E[C_{\lambda,\mu}]$	Expected number of re-key message per second, given arrival rate, λ, and average service rate, μ (msg/s)
T	Total traffic or network bandwidth used (bits/s)
σ	$\triangleq T/R$
k	Branching factor of a key tree

Poisson distribution peaks at its mean, we approximate the distribution as a δ-function at its mean, i.e., $\pi_i \approx \delta(i - \frac{\rho}{m})$, where

$$\delta(i - \frac{\rho}{m}) = \begin{cases} 1, & \text{if } i = \rho/m; \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

Therefore,

$$f(m) \approx \frac{\rho}{m}\sum_{i=0}^{\infty}\delta(i - \frac{\rho}{m})(E[J_i] + E[L_i]) \quad (7)$$

$$= \frac{\rho}{m}(E[J_{\frac{\rho}{m}}] + E[L_{\frac{\rho}{m}}]) \quad (8)$$

$$\approx \frac{\rho}{m}(2+k)\log_k\frac{\rho}{m}. \quad (9)$$

Therefore,

$$\sigma(m) \approx m\beta(\frac{\rho}{m})(2+k)\log_k\frac{\rho}{m} + m \quad (10)$$

$$\triangleq \hat{\sigma}(m), \quad (11)$$

and m^* can be obtained by setting $d\hat{\sigma}(m)/dm = 0$, i.e.,

$$m^* = \frac{\beta\rho(2+k)}{\ln k}. \quad (12)$$

Note that as $m^* \geq 1$ ρ should be greater than $\ln k/((2+k)\beta)$; otherwise, we should use a single server. Furthermore, since $\rho/m^* = \ln k/(\beta(2+k))$, the optimal group size is a constant.

III. Illustrative Numerical Examples and Results

In this section, we present some illustrative numerical results of the secure server network studied. Since the system parameter β has a determinant effect on the system performance (in terms of total bandwidth consumed), we first show its representative value for some multimedia applications given average user holding time ($1/\mu$) and data

TABLE II

SYSTEM PARAMETERS FOR DIFFERENT APPLICATIONS.

Application	Stock quote system			Internet radio	Near-CD quality MP3 audio	Internet Video
Average holding time	5 mins	30 mins (for part-time day trader)	4 hours (for investor or agent)	2 hours (~ 1 to 2 programs)	30 mins (>7 songs)	30 mins (CNN news)
μ (req./s)	1/300	1/1800	1/14400	1/7200	1/1800	1/1800
R (kbit/s)	5kb/s			16kb/s	96kb/s	256kb/s
β	1.33×10^{-3}	2.22×10^{-4}	2.78×10^{-5}	1.74×10^{-5}	1.16×10^{-5}	4.34×10^{-6}

rate of a stream (R) in Table II ($S = 2$ kbits). We see that β in reality is likely to range quite widely from 10^{-3} (stock quote systems) to 10^{-6} (video applications). In [3], it has been found that for a single server the optimal branching factor k for the key tree is around 4 independent of the number of users, which is also validated by us using analysis or simulation (results not shown here). Therefore, we will use $k = 4$ in our following study. We consider a baseline system with $\rho = 10^5$ and $\beta = 10^{-4}$, and vary them one at a time in our sensitivity study.

We first show the cost advantage in using a server network by plotting in Fig. 4 $\hat{\sigma}(m)/\hat{\sigma}(1)$ (i.e., the ratio of total traffic for a server network with m servers to a single server system) versus m given β ($\rho = 10^5$ and $k = 4$). The horizontal line corresponds to the single server case. For certain value of β (e.g., $\beta = 10^{-4}$), as m increases, $\hat{\sigma}(m)/\hat{\sigma}(1)$ first decreases gradually to reach a minimum (mainly due to the decrease in re-key messaging), and then increases steadily (mainly due to the increase in total data bandwidth required). There is hence an optimal m^* to minimize the total network bandwidth. From the figure, we see that "splitting" the server in an intelligent manner can substantially reduce the bandwidth requirement of the system. On the other hand, for some low value of β (e.g., $\beta = 10^{-6}$ in this case), $\hat{\sigma}(m)/\hat{\sigma}(1)$ monotonically increases, showing that serving the group of users with a single server is optimal. This is mainly because the data rate is relatively too high as compared to the re-key overhead to merit splitting. Therefore, the bandwidth saved for re-key messaging cannot mitigate the overhead of the increase in the extra multicast data traffic. There is hence a "breakeven" β at which splitting should be done. This is in fact given by Eq. (12) when $m^* = 1$, i.e., the breakeven β is at $\ln k/(\rho(2+k))$, which is equal to 2.3×10^{-6} for our baseline.

We have also compared our analysis with simulation in this figure. The discrete points represent our simulation results while the solid line represents our analysis. Clearly, our analysis matches very well with the simulation, showing the validity of our model. In the remainder of this section, we hence will use analysis in presenting our results.

We show in Fig. 5 m^* versus ρ using Eq. (12). Clearly it is a straight line. Given certain values of β and k, as the external arrival rate (and thereof the number of concurrent users) increases, the user pool should be split into more groups. The group number may range widely from a few to several hundreds. What worths noting from this result is that, as the underlying arrival rate changes, the number of users served by each server given by ρ/m^* should be kept constant ($= \ln k/(\beta(2+k))$, which is roughly equal to $2,300$ for the baseline) in order to minimize the overall network bandwidth. Therefore, the server network should execute some split-and-merge mechanism to dynamically adjust m to achieve such optimum, which we will discuss in the next section.

We finally explore the maximum reduction in network bandwidth of the server network as compared to the single server case. This is shown in Fig. 6 as the maximum bandwidth saving, defined as $1 - \hat{\sigma}(m^*)/\hat{\sigma}(1)$, plotted against β. As β increases, we tends to split the user pool into more groups and the saving therefore increases. The saving first increases slowly and then increases somewhat logarithmically with β. In other words, the saving increases with β with a decreasing rate. We see from the figure that with server networks, the network bandwidth requirement can be greatly reduced.

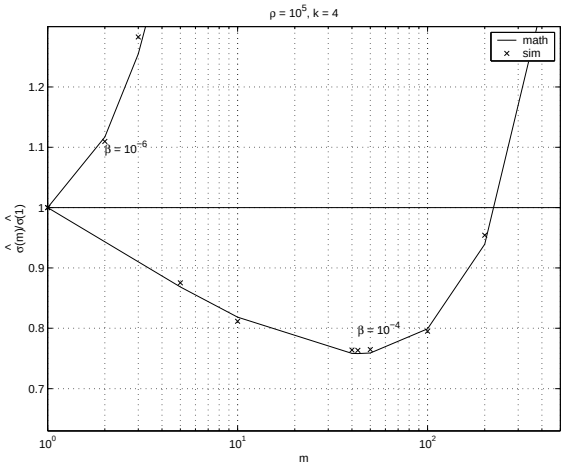

Fig. 4. $\hat{\sigma}(m)/\hat{\sigma}(1)$ vs. m, given β ($\rho = 10^5, k = 4$).

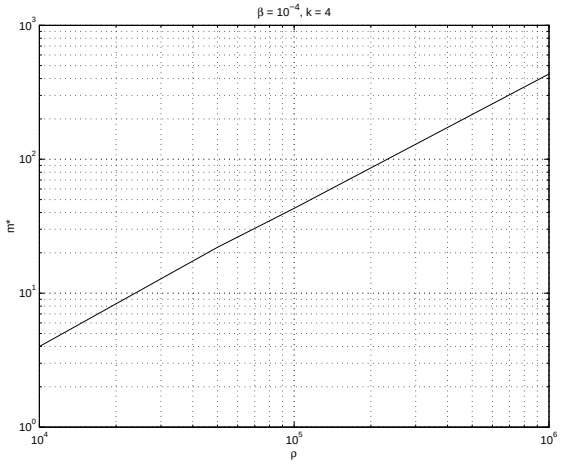

Fig. 5. m^* vs. ρ ($\beta = 10^{-4}, k = 4$).

IV. DYNAMIC MERGE-AND-SPLIT SCHEME

If the average number of concurrent users in a server is dynamic, then we need an advanced technique to adjust the number of groups in a physical server to be optimal.

Therefore, in this section, we introduce a simple and robust method to handle dynamic group merging and splitting which incurs negligible overhead and works well with a hierarchial key tree scheme as described in Sect. II-A.

Since m^* is directly proportional to ρ (see Fig. 5) for a given β. In general, the optimal number of users in a server should be a constant. Therefore, we can set a threshold on the maximum number of users in a sever, ϕ_{max}, before splitting is necessary and a threshold on the minimum number of users, ϕ_{min}, among two or more servers for which merging is necessary. If the number of users in a server increases and exceeds ϕ_{max}, we can split the user pool in this server into two or more. On the other hand, If the total number of users in

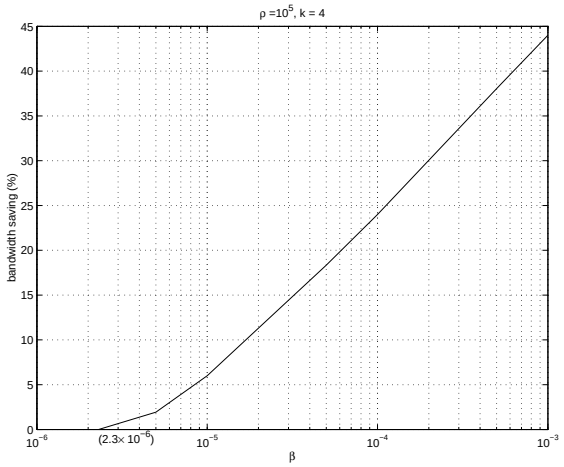

Fig. 6. Max. saving in bandwidth vs. β for a server network as compared to the single server case ($\rho = 10^5, k = 4$).

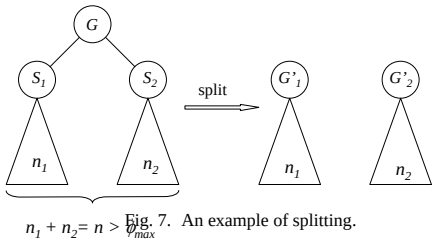

Fig. 7. An example of splitting.

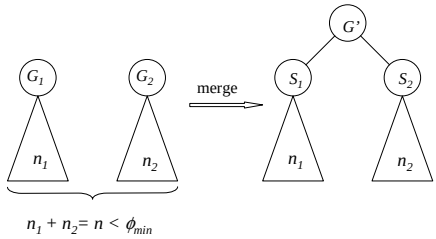

Fig. 8. An example of merging.

two or more servers decreases and becomes lower than ϕ_{min}, then we can merge these groups together. The total number of users of these merged groups, however, should not exceed ϕ_{max} after merging.

Let us describe and analyze the more general case of a k-ary tree. If the number of users in a server is greater than ϕ_{max}, we can split the user pool into k groups and set up some new logical servers to serve these new groups. To achieve splitting, we can simply remove the root node from the hierarchical key tree and form k new groups which are subgroups of the original key tree. Hence, the multicast data sent to the new k groups can be encrypted by their subgroup keys from their own servers. This is secure because these keys are only known by all members within their group. Since there is no new key that has to be generated and sent out, the cost of splitting is zero.

If there are k servers in which the total number of users is less than ϕ_{min}, we can merge the groups in these servers into an aggregated group served by only one of the logical servers. To achieve merging, we can add a new root node and the former root nodes of these groups become second level internal nodes. Since there is a new group key generated which has to be known by all the users in these groups, we can encrypt the new group key by each group's original group key and send it to each corresponding group. Therefore, for each merging of k groups, there are k overhead re-key messages which have to be sent out.

Figure 7 and 8 show an example of merging and splitting in a binary key tree. If there is a group in which the total number of users, n, is greater than ϕ_{max}, we can split the group into two and the original subgroup keys, S_1 and S_2 become the new group keys, G_1' and G_2', for these two new groups respectively. Whereas, if there are two groups in which n is less than ϕ_{min}, we can merge these two groups and generate a new group key. The original group keys, G_1 and G_2, become subgroup keys, S_1 and S_2, which can be used to encrypt the new group key, G' that is sent to these two groups. Hence, the new merged group will have two sets of message overhead, one for each subgroup.

V. Conclusions

Several key management schemes have been proposed in order to provide backward and forward secrecy in secure multicast systems. An efficient solution for key management is a hierarchical key tree approach, which reduces the number of re-key messages for each membership change to $O(\log N)$ where N is group size. However, most current approaches focus on a single server, which may cause higher overhead for a large number of users. In order to reduce the complexity and manageability of the single server system, we propose a distributed server network. In addition, we study the trade-off between re-key messaging and multicast data traffic in distributed server network to minimize the total traffic, generated by these traffic.

In this paper, we present a simple model to analyze this trade-off and found that there is an optimal number of logical servers (consisting of a data and control manager), m^*, proportional to the number of users in a system for a given average user service time, re-key message packet size and data traffic rate. The maximum gain of a system is influenced by the average number of users, the average user holding time, the size of the re-key message packet and the rate of multicast data traffic. When data traffic rate or average user holding time increases, maximum gain decreases; while the average number of users increases, the maximum gain increase. To tackle the problem of dynamic user traffic, we propose a scheme to split and merge the logical servers to further reduce the total traffic.

For some low bit rate applications such as stock quote systems, our example shows that up to 45% of total bandwidth can be saved. Therefore, distributed server networks are an efficient means to reduce total bandwidth when a trade-off between re-key messaging and multicast data traffic exists.

References

[1] S. E. Deering, "Multicast routing in internetworks and extended LANs," in *Proc. ACM SIGCOMM*, August 1988, pp. 55–64.

[2] W. Diffie, "Authenticated key exchange and secure interactive communication," in *Proc. SECURICOM*, 1990, pp. 300–306.

[3] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communication using key graphs," in *Proc. ACM SIGCOM*, 1998.

[4] D. M. Wallner, E. J. Harder, and R. C. Agee, "Key management for multicast: Issues and architectures," *RFC 2627*, June 1999.

[5] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 708–716, 1999.

[6] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, "Key management for secure Internet multicast using boolean function minimization techniques," *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings*, vol. 2, pp. 689–698, 1999.

[7] R. Canetti, T. Malkin, and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption," *Advances in Cryptology - EUROCRYPT '99. International Conference on the Theory and Applications of Cryptographic Techniques. Proceedings*, pp. 459–474, 1999.