Key Management for 3G MBMS Security

Wenyuan Xu, Wade Trappe, Sanjoy Paul*
Wireless Information Network Laboratory (WINLAB)
Rutgers University, 73 Brett Rd., Piscataway, NJ 08854.

***Abstract*— *Multimedia Broadcast/Multicast Service*(MBMS), which increases network efficiency, and facilitates group-oriented applications, is a service in 3G networks that is about to be standardized. A key security issue for MBMS is the design of a mechanism that controls access to multicast data. Traditionally, access control is done by distributing and maintaining key information. However, the existing architecture of mobile equipment prevents keys from being stored securely on the user side, making the traditional key management insufficient to protect the confidentiality of multicast data. Some discussion in 3GPP has addressed the issue, but the existing scheme achieves the security goal at the cost of high bandwidth overhead, unpredictable peak bandwidth and potential server implosion problems. Currently, no quantified convincing research has been done to study these issues. In this paper, we present new schemes that solve the problems, and quantify their scalability and bandwidth overhead.**

I. Introduction

Third generation wireless networks are being deployed to replace existing second-generation wireless networks. These new networks will allow users to access services that are currently available on conventional wireline networks. One popular class of applications that will be deployed on future wireless networks is the group-oriented application, such as pay-per-view broadcasts or communal gaming. In conventional IP networks, the use of multicast networking is most appropriate for group-oriented applications since multicasting exploits the distribution of group members in the network to minimize packet duplication. The use of multicast in 3G networks, however, is at an early stage of development. In fact, it was not until Release 6, in 2002, that the 3rd Generation Partnership Project (3GPP) provided the first technical specification of the *Multimedia Broadcast/Multicast Service* (MBMS) for 3G wireless networks[1], and later some parallel works appeared in 3GPP2[2]. The specification for MBMS will likely evolve over the next few years.

As multicast functionality is integrated into future wireless networks, the adaptation of multicast into commercial wireless applications will rely upon the ability for the service provider to control access to multicast data. The problem of controlling access to multicast data requires the distribution and maintenance of the *session key* (SK), the key used to encrypt the multicast content. The distribution of SKs, however, is not a sufficient mechanism to protect the confidentiality of multicast content. In fact, since 3G Mobile Equipments (ME) are not secure storage devices, it is possible for an adversarial user to read the ME memory, access the SK, and transmit the SK needed for unauthorized users to decrypt multicast content.

Recently, a security framework for addressing the issue of insecure ME storage for multicast was presented in the 3GPP forum[3]. This framework, which we shall refer to as S3-030040, attempts to make it cost-prohibitive for an adversary to distribute key information to unauthorized users. Although S3-030040 provides a sufficient deterrent for adversaries, it also introduces several drawbacks, such as high peak-to-average bandwidth, non-scalability and a network implosion problem.

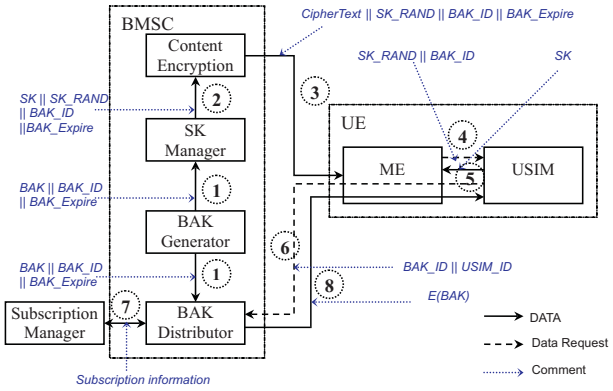
In this paper we present a new security framework that extends S3-030040 and addresses the weaknesses in S3-030040. Our proposed security framework consumes modest amounts of network resources and maintains a relatively constant bandwidth usage while achieving a higher level of system security. The paper is organized as follows: in Section II we give an overview of the insecure storage problem in 3G networks and describe the S3-030040 framework. In Section III, we introduce several solutions for reducing the bandwidth associated with control information These approaches are used as building blocks for our security framework. We present our simulation model and experimental results in Section IV. Finally, we present conclusions in Section V.

II. MBMS Architecture and Multicast Security

The insecure storage problem is caused by the internal structure of the User Equipment, which is composed of the Mobile Equipment (ME), such as the cellular phone or PDA, and the User Services Identity Module (USIM), also known as SIM card[4]. The USIM is not powerful enough to decrypt bulk amounts of data, and therefore the ME must decrypt encrypted content sent by the Broadcast/Multicast Switch Center (BMSC). As a result, the decryption keys have to be stored in the ME. Unfortunately, ME is not a secure storage device. If the session key, which controls access to group messages and is shared by all the group members, is stored in the ME, it is possible for the subscribers to distribute it to non-paying users, thereby allowing non-paying users unauthorized access to the group communication. Making the memory in the ME secure is too costly to be practical, and thus other approaches are needed to solve the insecure storage problem. The S3-030040 [3] solution was presented to address the secure storage problem.

We now summarize the S3-030040 approach. Assume the ME stores the session key and performs decryption. An attack starts when a legitimate user A accesses the session key and illegally distributes it to a non-paying user B. User A must send out the session key by unicasting a short message or making phone call. If we assume that there is a cost per message sent, as is common in current pricing models for wireless networks, and that the session keys change frequently, then the cost of distributing and receiving the leaked keys will ultimately exceed the cost of subscribing to the multicast service. As a result, users will be less inclined to mount an attack on the keys.

*The authors may be contacted by e-mail at wenyuan, trappe@winlab.rutgers.edu, sanjoy@research.bell-labs.com



1: BAK Generator generates a new *BAK* and sends it to SK manager and BAK distributor.
2: SK manager randomly selects a number *SK_RAND*, gets *SK = f (SK_RAND, BAK)* and sends *SK_RAND, SK, BAK_ID,* and *BAK_Expire* to Content Encryption (CE).
3: CE encrypts the content using *SK*, attaches *SK_RAND, BAK_ID,* and *BAK_Expire* as plaintext to the packet. And it sends out the packet to ME.
4: If no new *SK* is used, ME just decrypt the packet. Otherwise, ME sends a request containing *SK_RAND* and *BAK_ID* to the USIM.
5: If USIM knows the BAK indexed by *BAK_ID,* it calculates *SK = f (SK_RAND, BAK)* and sends *SK* to ME. Otherwise, step 6, 7 and 8 are needed before USIM can send *SK* to ME.
6: USIM sends a *BAK* request including its identification to BAK distributor.
7: BAK Distributor checks the USIM legality with Subscription Manager (SM), and receives the USIM's Registration Key (RK) from SM, which may have been built into USIM card at the factory.
8: BAK distributor sends out the encrypted *BAK* to the USIM using Temporary keys derived from RK.

Fig. 1. The S3-030040 MBMS Security Framework.

In order to change the session keys frequently, while maintaining a reasonable amount of communication overhead, a key hierarchy was proposed in S3-030040. First, there are *registration keys* (RK), *temporary keys* (TK) and *broadcast access keys* (BAK), which are known only to USIM and BMSC but not to the ME. Second, the Session Key (SK), which is also known as the Short-Term Key, needs to be changed at a high rate and has a shorter life span than the BAK. To facilitate the distribution of the keying material, the S3-030040 security framework is composed of the BAK Distributor, BAK Generator, Short-term Key Manager, Content Encryption entity and Subscription Manager. All entities except the Subscription Manager are located in the BMSC. The dataflow among these entities is shown in Fig. 1.

Note that ME doesn't contain the current BAK_j but only has the current SK_i. In order to attack, every SK_i has to be sent out and the cost is determined by the SK change rate and price per transmission. Therefore, by adjusting the SK change rate the security goal can be easily achieved. Further, as Fig. 2 shows, only when the user knows both SK_RAND_i and BAK_j can he/she get SK_i and decrypt the ciphertext.

The above framework solves the insecure storage problem, but introduces the following drawbacks:

1. The SK_RAND is 32 bits and the SK is 128bits. The key space, and hence security, has decreased from 2^{128} to 2^{32}.

2. If the SK_RAND attached in each packet is compromised, even if the payload itself is correctly transmitted, a retransmission of the whole package is needed.

3. Transmitting SK_RAND introduces additional communication overhead.

4. When a new BAK is used, each USIM sends a BAK request to BAK Distributor leading to a *BAK implosion* problem. In addition, the point-to-point sending of BAKs generated as a result of these BAK requests significantly increases bandwidth usage.

5. The point-to-point BAK update scheme requires the server to process each BAK request individually, encrypt the new BAK for each client and transmit it to each client. This is clearly not scalable.

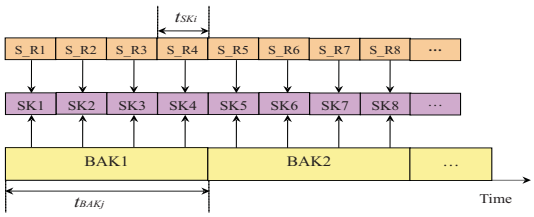


Fig. 2. The key hierarchy. S_R_i denotes SK_RAND_i. t_{SK_i} is the valid time slot for SK_i. t_{BAK_j} is the valid time slot for BAK_j, where $t_{SK_i} \ll t_{BAK_j}$.

III. An Improved MBMS Secure Framework

A. SK Generation Improvement

The first three drawbacks mentioned above are a consequence of the overhead associated with sending SK_RAND with the message over the wireless channel. In order to reduce this overhead, we propose a new method for generating the session keys that does not involve sending SK_RAND. Instead of updating SK by sending out SK_RAND, we send out an ID to indicate which SK is being used. Further, each time a new SK is used, the USIM generates the SK locally by employing a function chain consisting of cryptographic one-way functions. As a result, if we use 8 bits to transmit the ID instead of a 32-bit SK_RAND, we can decrease communication cost in each packet by 24 bits while increase the key space to 2^{128} from 2^{32}. The SK chain can be generated by: $SK_0 = f(SK\_SEED, BAK)$, $SK_1 = f(SK_0, BAK)$, ..., $SK_{i+1} = f(SK_i, BAK)$.

Here f is a suitable cryptographic one-way function, such as SHA-1 or MD5. The one-way property guarantees that an adversary cannot calculate BAK, even if he/she knows SK_i, SK_{i+1} and f. It is desirable that f is computationally efficient since the USIM has limited computational power.

By generating the chain of SKs locally, the success of generating a new SK is no longer as closely related to the information sent *in* each packet as in the S3-030040 method. Instead, all that is needed is the ID, which is usually sequential and can be inferred even if the ID portion of the message is corrupted. This fact provides an additional advantage over S3-030040 for cases in which a rekeying instance is missed by a user. Rather than requesting retransmission of prior rekeying packets, the user can locally reconstruct missed session keys in the key chain. Finally, since the function f can be the same as that proposed in S3-030040, the improved SK-updating scheme has at least the same level of system security as that of S3-030040 method, but at the cost of lower bandwidth usage.

B. BAK Distribution Improvement

We now describe improved methods for distributing the BAK. We begin by discussing the issue of *BAK Update Initiation*. When a new BAK is used, each USIM will simultaneously send a BAK request to BAK Distributor. There is no mechanism to coordinate BAK requests. If the group size is big, then this will cause a serious *BAK implosion* problem.

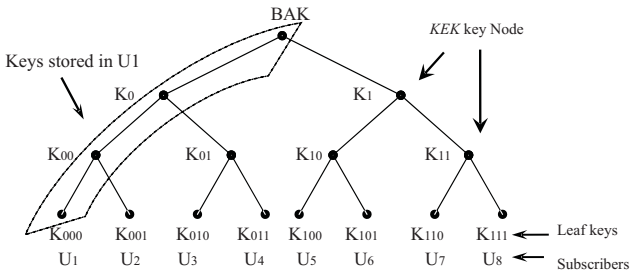


Fig. 3. The BAK key tree with eight subscribers.

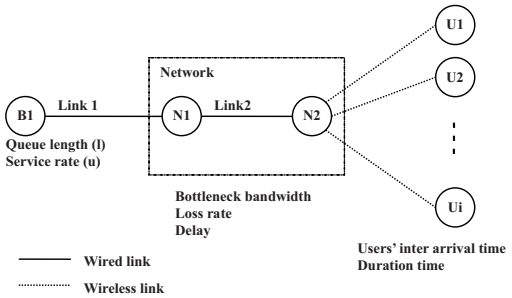


Fig. 4. The simulation topology.

We observe that the server precisely knows when a new BAK is used and as a result, also knows that it will receive a large number of BAK request messages. Based on this knowledge, the server can proactively push the new BAK to each subscriber. As a result of pushing instead of pulling, the implosion problem is eliminated. One advantage of this scheme is that it allows a fine-grained billing: instead of billing by number of BAKs used as in S3-030040, we can now bill by the duration a user participates in a multicast service.

Next, we discuss the issue of *BAK Distribution*. To maintain a high security level, the BAK needs to be changed and distributed to each subscriber securely whenever users join or leave the group. If we simply change the pull scheme into a push scheme, then the communication cost and BAK encryption cost will be proportional to the multicast group size n. This is inefficient for large groups.

In order to obtain a more scalable solution to BAK distribution, we propose to use a key tree [5][6] consisting of intermediate keys known as Key Encryption Keys (KEK) and a root key, BAK. These KEKs are arranged in a hierarchy and assist in updating the root key BAK in subgroups. For example, Fig. 3 depicts a binary tree with 8 subscribers U_1 to U_8, who have already joined the group. When some users join the group, the old BAK is used to encrypt and distribute the new BAK. The cost for this is only 1 unit time for encryption and 1 packet for communication.

When a user leaves, only the KEKs known to the leaving user need to be changed. In this case, the lower level old KEK is used to encrypt the higher level KEK. For example, if U_1 leaves, then BAK, K_0 and K_{00} should be changed. We use $\{BAK'\}_{K_1}$ to represent the message with content BAK' encrypted by K_1. In order to update all the KEKs the following messages need to be sent out by the BAK Manager (denoted by s in the following lines):

$$\begin{aligned} s \rightarrow \{U_2\} &: \{K'_{00}\}_{K_{001}}, \{K'_0\}_{K'_{00}} \\ s \rightarrow \{U_3, U_4\} &: \{K'_0\}_{K_{01}} \\ s \rightarrow \{U_2, U_3, U_4\} &: \{BAK'\}_{K_{0'}} \\ s \rightarrow \{U_5, U_6, U_7, U_8\} &: \{BAK'\}_{K_1} \end{aligned}$$

The communication cost and BAK encryption cost of using the key-tree updating method is at most $O(log(n))$ in the leave case. The cost of BAK update in the case of join, and in the case of join/no-leave is $O(1)$.

The key tree efficiently updates the BAK for each join/leave, but when the frequency of the user join/leave events is high, the $BAKs$ still have to be changed frequently. As a result, the bandwidth used in key update can still be considerably large. This is the trade off between security and communication cost. If we only want the same security level in S3-030040 proposal, we need only update the BAK periodically no matter whether the subscribers have changed or not. But if we want higher security levels, we need to change the BAK whenever users join or leave the group. To achieve optimal tradeoff, we can batch join/leave events that occur during a suitably chosen window of time into one rekeying event instead of handling them separately. Finally, to further achieve communication efficiency, we can map the key tree to the physical location of the user, and as a result the key updating messages will be transmitted using as few links as possible[7].

IV. Simulation

We now present simulations to validate the claimed improvements provided by the proposed key distribution scheme. We begin by discussing the simulation model, then describe the models used for the data traffic and the membership dynamics, and finally present the simulation results.

A. Simulation Model

Our simulations were performed in NS-2. The simulation model consists of three separate components, as depicted in Fig. 4. The first entity is the BMSC, depicted as node B_1. The parameters for the BMSC include the length of the join/leave request queue, and the service time for these requests. The Network component consists of two internal nodes N_1 and N_2, as well as $Link_2$. The nodes N_1 and N_2 correspond to the endpoint nodes of the bottleneck link $Link_2$. Only two nodes are used to represent the Network since we are primarily concerned with capturing the bottleneck effect in the Network. As an example, node N_1 could correspond to the *Gateway GPRS(General Packet Radio Service) Supporting Node*(GGSN), node N_2 the *Serving GPRS Support Node* (SGSN), and $Link_2$ the bottleneck link between the GGSN and the SGSN. The parameters associated with $Link_2$ are the link bandwidth, link delay and loss rate. Finally, in the third part, we represent the mobile users $U_1, U_2 \ldots U_i$. The mobile users are characterized by the rate at which new users join the multicast group (referred to as arrival rate), and the duration of their service.

Our simulation models the join/leave patterns for the mobile users under different service scenarios including a chatting service and a multimedia service.We used UDP to transmit the data and control packets, and we developed an MBMS application

on top of UDP. The MBMS application emulates the functions belonging to the Content Encryption Entity, the SK-Manager, the BAK Distributor, the BAK Generator and the Subscription Manager. By designing the model in this manner, we have kept the simulation simple enough to control while also maintaining enough complexity to capture realistic scenarios.

B. Traffic Model and User Join/Leave Model

We are not aware of any existing research on data traffic for 3G multicast. On the other hand, IP multicast is a mature research field and there is a significant body of research on IP multicast traffic measurement and the join/leave behavior of IP multicast group members[8]. For the purpose of this paper, we assume that the characteristics of 3G multicast applications are similar to IP multicast applications, and therefore we will use IP multicast traffic patterns in our 3G experiments.

The majority of IP multicast traffic depends on only a small set of "special" events, such as a single Internet-wide multicast of a seminar. Therefore, in our simulations, we only considered one multicast service running on the 3G network, and studied two different application scenarios. First, we simulated a multicast movie session, involving multicasting of video and audio data, lasting for 4000 seconds. Second we simulated a chat-session, involving textual data only, lasting for half a day.

1) Movie Multicast Session: The multicast movie session is a pay-per-view service in which the user is charged for the amount of the movie he watches. Since advanced compression codecs will most likely be used on 3G and beyond cellular systems, we chose to use MPEG-4 and H.263 video traces, and statistical data of *Start Wars IV, Jurassic Park I and Silence of the Lambs* from [9]. Frame period of $t = 40ms$, per-user bandwidth of 50kbps - 80kbps, and a mean frame size of $x = 0.5KBytes$ were used. Using the frame size histograms presented in [9], the frame size of *Star Wars IV* can be adequately modeled using a lognormal distribution. We therefore generated $N = 100,000$ frames with a frame size according to the lognormal distribution with the mean $\mu = 0.27$, $\mu = 0.77$ and $\mu = 0.53$ KBytes respectively for each of the three movies.

The inter-arrival time and the membership duration characterize the group join/leave behavior. According to the model [8], the user join/leave behavior may be modeled as a Poisson Process. The inter-arrival time was modeled using an exponential distribution with mean of $1.5s$ for the first 150 seconds of the movie, and an exponential distribution with mean of $150s$ for the remainder of the movie. Using real measurements of the UCB seminar in[8], the mean membership duration was chosen to be 46 minutes. In summary, our join/leave process exhibits a larger fraction of joins near the program's scheduled start, and fewer joins after the first few minutes.

2) Chat Session: The traffic of a text-only chat session is one of the least investigated types of traffic. In order to model it, we use the exponential distribution to represent inter-packet arrival times, and assume the mean inter-packet arrival time is related to the group size. Additionally, we assume the mean inter-packet arrival time is much larger than for a movie since humans are relatively slow machines for generating traffic. The packet size is usually short since people tend to send simple message like "hello". To model the user join/leave behavior,

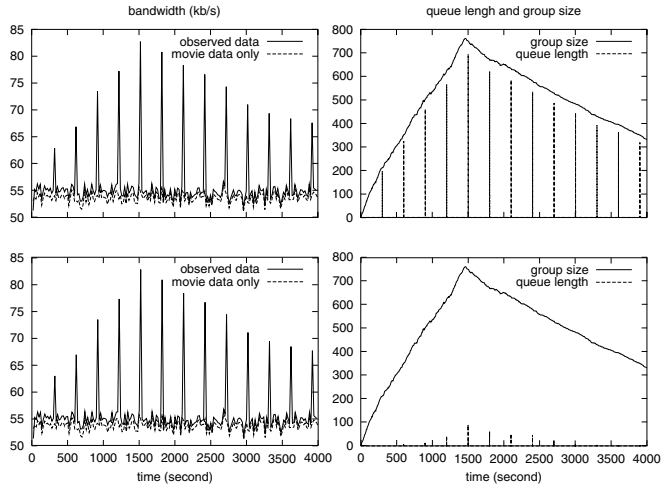


Fig. 5. The bandwidth and queue length result in the two pulling modes.

we chose to characterize a chat session using the long session model described in [8]. This was done since chat services typically run for periods on the order of a day. As a result, the session duration is modeled as a Zipf Random Variable [8], while the user inter-arrival time is an exponential Random Variable.

We chose to simulate two scenarios: a weekday and a weeknight chat session. A weekday chat session was simulated with a mean inter-arrival time of 22.1 seconds, and mean duration of 5 hours. These values were taken from the measurements of the STS-63 session that were published in [8]. A weeknight chat session was simulated with a mean inter-arrival time of 4 minutes and a duration of 6 hours.

C. Simulation Results

We ran our simulations for six different modes of operation:

1. (o mode): *o* mode corresponds to the original S3-030040 mode. In this mode, SKs are generated from SK_RAND. Users send BAK requests to the server whenever they need a new BAK. The server distributes the requested BAKs using unicast. Each transmission is encrypted with each user's TK.

2. (or mode): In *or* (original randomized) mode, users still pull BAKs from the server and calculate SKs from SK_RAND. However, to partially alleviate the BAK implosion problem, users send out their BAK requests after a random amount of time $(uniform(0,1)sec)$ following the moment they detect they need a new BAK.

3. (npr mode): *npr* denotes a new point-to-point real-time mode. The SKs are generated using a one-way function. The server pushes the new BAKs to each user via unicast whenever the server receives a join/leave request.

4. (npc mode): *npc* mode is similar to *npr* mode, except that the server updates the BAK after a threshold amount of time after a new BAK is used. In our simulation, the time threshold is set to be the same as the BAK expiration time in *o* mode.

5. (ntr mode): *ntr* denotes the new tree-based real-time mode. In this mode, the server pushes the new BAKs to each user when a new join/leave request is received. The server distributes the BAKs using a tree-based key update scheme.

6. (ntc mode): *ntc* mode is almost the same as *ntr* mode. The only difference is that the BAK is updated after it is used for a threshold amount of time.

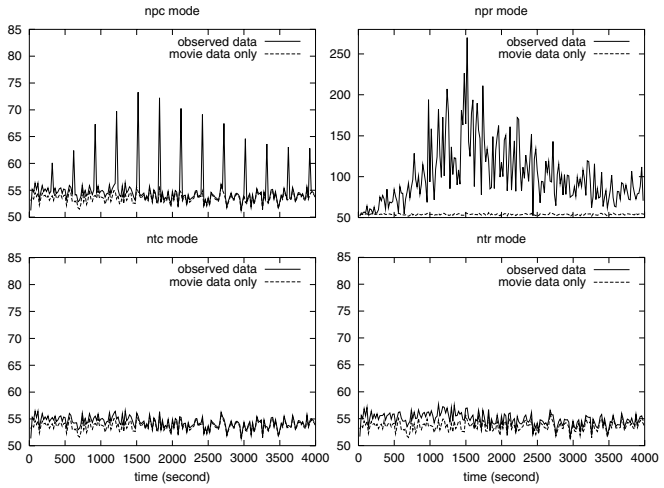


Fig. 6. The bandwidth (kb/s) in the four pushing modes.

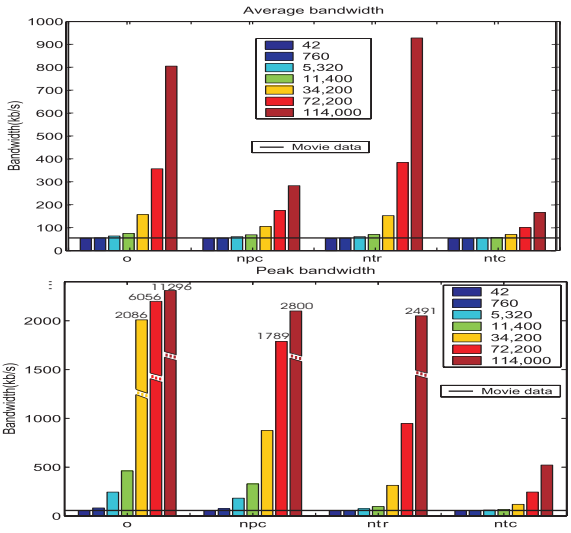


Fig. 7. The bandwidth for different group sizes and modes.

Fig. 5 is the simulation results for the queue and bandwidth data in the first 2 pulling modes. The input traffic pattern is the low resolution *Star War IV*. From Fig. 5, we note that there are two problems in *o* mode: (1) the queue length is proportional to the group size at the moment a new BAK is used in *o* mode, and (2)the peak bandwidth is very high. The *or* mode led to much shorter queues than in *o* mode. This implies that sending out the BAK request after a random amount of time does alleviate the BAK implosion problem. But, it does not help the high peak bandwidth problem.

Fig. 6 presents the simulation results for the bandwidth used in the four pushing modes. The input traffic pattern is the same as in Fig. 5. For all four modes, the queue length is not an issue, so we do not present the queue length in the figure. For point-to-point server pushing BAK modes (*npc, npr*), the peak bandwidth is still a problem. This is especially true in *npr* where too much bandwidth is devoted to update the new BAK during a change in the group membership. The peak bandwidth of *ntr* and *ntc* is the lowest.

To test the scalability of the different modes, we ran the simulations for different group sizes ranging from 42 to $114,000$. A group size of $114,000$ can occur when a carrier with a million users has 1% of its users subscribe to a popular event, like the Olympics. The first plot in Fig.7 shows the average bandwidth usage over the entire movie. We observe that *ntc* has the best scalability in terms of average bandwidth. But when group size is less than $11,400$, the benefit of *ntc* for average bandwidth reduction is less pronounced. However, average bandwidth, which is partially determined by the BAK update rate, is not enough for judging scalability. The peak bandwidth is more significant as traffic bursts are problematic for networks. The second plot in Fig.7 depicts the peak bandwidth used, which confirms that *ntc* is the most scalable mode. As the group size becomes larger than 5000, the maximum bandwidth overhead of *ntc* scheme is 96% less than that introduced by the S3-030040 scheme. Similar results were observed for the *chat session*.

V. Conclusion

Before MBMS can be successfully deployed, the problem of controlling access to multicast data has to be solved. The insecure storage problem of the existing ME reduces the effectiveness of traditional multicast key distribution mechanisms. One approach to solve this is to change the SK frequently enough to make an attack too costly to perform. Previous methods using this strategy have significantly increased load on the network. We have presented techniques that eliminate the BAK implosion problem and also reduce the bandwidth need to perform key management. We presented simulation results for a server-based multicast video service to show that from the system performance and scalability point-of-view, the best security framework is the one where the servers push the new BAK to group of users based on a key-tree structure, and use time thresholding together with the join/leave request as the indicator of when to update a new BAK. On the other hand, if the simplicity of implementation is the major concern and if the group size is always small, it might be better to implement a server-pushing point-to-point BAK update scheme, since the benefits are less-pronounced when the group size is less than 760 users.

References

[1] 3rd Generation Partnership Project, "Multimedia broadcast/multicast service; release 6," Tech. Rep. TS 22.146 V6.1.0, Sep. 2002.

[2] P. Agrawal, M. C. Chuah, and J. Zander, "Multimedia multicast/broadcast services in 3G/4G networks," *Communications Magazine, IEEE*, vol. 42, no. 2, 2004.

[3] P. Hawkes and R. Subramanian, "Explanation of BAK-based key management," Tech. Rep. 3GPP S3-030040, QUACOMM, France, Fre. 2003.

[4] 3rd Generation Partnership Project, "Network architecture; release 5," Tech. Rep. 3GPP TS 23.002 V5.9.0, Dec. 2002.

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

[6] C. K. Wong, M.Gouda, and S.S. Lam, "Secure group communications using key graphs," *IEEE/ACM Transactions on Networking*, vol. 8, no. 1, pp. 16–31, February 2000.

[7] Y. Sun, W. Trappe, and K.J.R. Liu, "An efficient key management scheme for secure wireless multicast," in *IEEE Int. Conference on Communications*, 2002, pp. 1236–1240.

[8] K. Almeroth and M. Ammar, "Collecting and modeling of the join/leave behavior of multicast group members in the mbone," *High Performance Distributed Computing Focus Workshop (HPDC)*, August 1996.

[9] F.H.P. Fitzek and M. Reisslein, "MPEG-4 and H.263 video traces for network performance evaluation," *IEEE Network*, vol. 15, no. 6, pp. 40–54, November/December 2001.

[10] 3rd Generation Partnership Project, "General packet radio service (GPRS); service description; release 5;," Tech. Rep. TS 23.060 V5.4.0, Dec. 2002.