A block-free TGDH key agreement protocol for secure group communications

Xukai Zou
Department of Computer and Information Sciences
Indiana University - Purdue University Indianapolis, USA
Email: xkzou@cs.iupui.edu
Byrav Ramamurthy
Department of Computer Science and Engineering
University of Nebraska-Lincoln, USA
Email: byrav@csce.unl.edu

Abstract: Secure group communication (SGC) has been an active research area because several applications require it. In this paper, we propose a Block-Free Group Tree-based Diffie-Hellman (BF-TGDH) key agreement protocol, based on a recently proposed Group Tree-based Diffie-Hellman (TGDH) key agreement protocol, for secure group communication. The new protocol has the following specific properties: (1) no blocking during the rekeying process and seamless group communication without any interruption; (2) resistance to the Man-in-the-Middle attack; (3) authentication of message senders using inherent (ElGamal) signature protocol.

Keywords: System design, Network measurements, Experimentation with networks, Secure group communication, key management, Diffie-Hellman key exchange, contributory key agreement.

I. INTRODUCTION

Secure group communications (SGC) refers to a setting in which a group of participants can send and receive messages to group members in a way that outsiders are unable to glean any information even when they are able to intercept the messages. SGC is becoming an extremely important research area because many applications require it including teleconferencing, tele-medicine, real-time information services, distributed interactive simulations, collaborative work, interactive games and the deployment of VPN (Virtual Private Networks).

Secure group communication can be typically classified as one-to-many (broadcast) communication, few-to-many communication and many-to-many (peer) communication [1]. In peer group communication, all group members can send and receive messages destined to the group. The primary problem in secure group communication is group key management. There are two typical categories of key management protocols for peer group communication. One is centralized key distribution and the other is distributed (contributory) key agreement. Among all key management protocols, the key tree scheme [2], [3], [4], [5] is a very powerful approach which can be used for both one-to-many communication as well as many-to-many communication. The key tree scheme can be used for centralized key distribution as well as for distributed (contributory) key agreement. In general, the contributory key agreement protocols are primarily different variations of the n-party Diffie-Hellman key exchange [6], [7], [8], [9], [10], [11].

Tree-based Group Diffie-Hellman (TGDH) contributory key agreement protocol [8] is robust and efficient in the sense that it can deal with network partition and that the number of rounds for rekeying is limited by $O(log_2 n)$ where n is the number of members currently in the group. However, with this protocol (also in other protocols), during the period of rekeying (which occurs whenever member(s) join or leave the group), all group members stop data communication and wait until the new group key is formed (in a distributed manner). This will cause frequent interruptions of group communication. In case a rekeying packet is delayed or lost, the intervals (latency) and frequencies of interruptions may become annoying. The latency problem during initial key establishment and rekeying has been noticed by researchers. In paper [12], the authors considered the situation where the group members have different communication delays and computation capabilities and proposed an approach which tries to set the members with similar delays and capabilities in the neighboring positions (on the virtual key tree), thus reducing the latency of group key management.

Based on TGDH, we propose a new protocol which is called Block-Free Tree-based Group Diffie-Hellman key agreement protocol (BF-TGDH). By introducing dummy

members, the protocol deals with the key management (rekeying process) as a background operation and group communication can continue seamlessly during the entire period of group communication regardless of joining and leaving of members, even when multiple members join and/or leave at the same time. Moreover, the scheme resists the Man-in-the-Middle attack, which exists in TGDH and provides authentication of message senders using inherent (ElGamal) signature instead of an external RSA signature as in TGDH.

The paper is organized as follows. In the next section, we briefly describe the TGDH protocol. Then we present our new Block-Free TGDH protocol in section III. The issues of performance and security are discussed in section IV. We conclude the paper in section V.

II. Tree-based Group Diffie-Hellman (TGDH) key agreement protocol

In this section, we briefly describe the TGDH protocol [8] since our new protocol is based on it. First, we summarize the Discrete Logarithmic Problem (DLP) and Diffie-Hellman key exchange principle since they form the basis of TGDH.

Discrete Logarithmic Problem (DLP) can be stated as follows: suppose p is a large prime and g is a generator for Z_p^*. Given $b \in Z_p^*$, it is difficult to find a such that $b = g^a \ mod \ p$. Two party Diffie-Hellman key exchange can be stated as follows: two members such as M_1 (with a private component a_1 and a public component $b_1 = g^{a_1} \ mod \ p$) and M_2 (with a private component a_2 and a public component $b_2 = g^{a_2} \ mod \ p$) can compute a shared key $SK = g^{a_1 a_2} \ mod \ p$ independently (called Diffie-Hellman key) without exposing private components a_1 and a_2. But any individual other than M_1 or M_2 cannot compute SK even if he/she knows the public components b_1 and b_2. It is well-known that the main problem with Diffie-Hellman key exchange is the Man-in-the-Middle attack. In TGDH (other n-party Diffie-Hellman key agreement protocols as well), both p and g are publicly known. Moreover, we call a_1 (similarly a_2) the (private) share of M_1 (similarly M_2) and, correspondingly, b_1 (similarly b_2) is called the *disguised public share* of M_1 (similarly M_2).

In TGDH, all members maintain an identical virtual binary key tree which may be balanced or not. The nodes in the key tree are denoted by $< l, v >$ where l is the level in the tree and v indicates which node in level l ($0 \leq v \leq 2^l - 1$ since level l hosts at most 2^l nodes). The root node is labeled as $< 0, 0 >$ and the two children of a (parent) node $< l, v >$ are labeled as $< l + 1, 2v >$ and $< l + 1, 2v + 1 >$. Every node

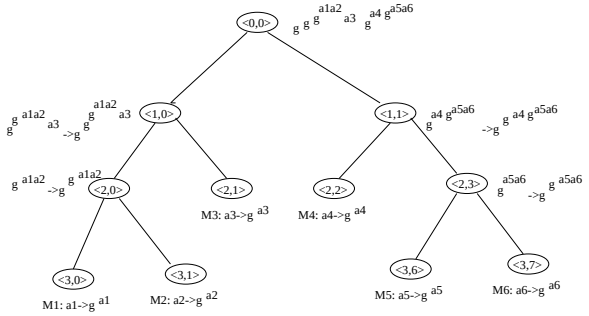

Fig. 1. Tree-based Group Diffi e-Hellman key agreement. The notation $k \rightarrow g^k$ means that members compute the secret key k, then compute the blinded key $BK = g^k$ and broadcast BK.

$< l, v >$ except the root node is associated with a secret key $K_{<l,v>}$ and a blinded (bkey) key $BK_{<l,v>} = f(K_{<l,v>})$ where $f(x) = g^x \ mod \ p$. The key $K_{<l,v>}$ of a (parent) node is the Diffie-Hellman key of its two children, i.e., $K_{<l,v>} = g^{K_{<l+1,2v>} K_{<l+1,2v+1>}} \ mod \ p$ ($=f(K_{<l+1,2v>} K_{<l+1,2v+1>})$), that is to say that the secret key $K_{<l,v>}$ of a node is formed from the secret key of one of its two children and the blinded key of the other child by using Diffie-Hellman key exchange protocol. Every blinded key $BK_{<l,v>}$ is publicly broadcast by some member(s) to the group. All group members are hosted on leaf nodes respectively. Every member M_i will select its own Diffie-Hellman private share a_i and compute the corresponding Diffie-Hellman disguised public share b_i. Suppose a leaf node $< l, v >$ hosts a member M_i, then the secret key $K_{<l,v>}$ of the leaf node is a_i and the blinded key $BK_{<l,v>}$ ($=g^{K_{<l,v>}}$) is b_i. Member M_i can compute all secret keys from its leaf node $< l, v >$ to the root node $< 0, 0 >$ (and of course, M_i will receive all blinded keys in the key tree). The root node has the secret key $K_{<0,0>}$ (there is no need for $BK_{<0,0>}$). The group data encryption key DEK is obtained by passing $K_{<0,0>}$ through a one-way function.

Figure 1 illustrates a key tree with six members. As shown in the figure, Node $< 3, 0 >$ hosts member M_2 and is associated with M_2's private share a_2 and disguised public share $b_2 = g^{a_2}$. M_2 can compute $K_{<2,0>} = g^{K_{<3,0>} K_{<3,1>}} = g^{a_1 a_2} \ mod \ p$ after receiving $b_1 = g^{a_1}$ and compute and broadcast $BK_{<2,0>} = g^{g^{a_1 a_2}}$. M_2 can continue to compute $K_{<1,0>} = g^{K_{<2,0>} K_{<2,1>}} \ mod \ p = g^{g^{a_1 a_1 a_3}} \ mod \ p$ when receiving $BK_{<2,1>} = g^{a_3} \ mod \ p$ from M_3 and then compute and broadcast $BK_{<1,0>} = g^{K_{<1,0>}} \ mod \ p. = g^{g^{g^{a_1 a_1 a_3}}} \ mod \ p$. Finally M_2 computes $K_{<0,0>} = g^{g^{g^{a_1 a_2 a_3}} g^{a_4 g^{a_6 a_7}}} \ mod \ p$ after receiving $BK_{<1,1>}$. It can be seen that the root key $K_{<0,0>}$ contains Diffie-Hellman private shares of all members; that is to say that the group key is obtained by the uniform contribution of all group members.

The join and leave operations are summarized below. A *sponsor* is a member and is defined as follows: the sponsor of a subtree is the member hosted on the rightmost leaf in the subtree and the sponsor of a leaf node is the member hosted on the rightmost leaf node (other than itself) of the lowest subtree this leaf node belongs to.

As indicated in the above example (for M_2), the initial computation of the root key is performed in multiple rounds by all members: every member selects its private share as the secret key and broadcasts the blinded key. Then along path from its leaf node to the root, every member computes the secret key of a node while receiving the blinded key from its sibling and computes the blinded key and broadcasts the blinded key.

When a member requests to join (the member will broadcast his/her disguised public share), every member can decide the insertion location for the joining member and determine the sponsor for the joining member. Every member updates the key tree by adding a new member node and a new internal node, and removes all secret keys and blinded keys from the leaf node related to the sponsor and the root node. The sponsor generates his/her new private share and computes all secret keys and blinded keys from his/her leaf node to the root node, and then broadcasts all new blinded keys. Every member computes the new root key after receiving the new blinded keys. It can be seen that the root key is computed in one round. Similarly the root key is computed in one round for single leave.

As for multiple leaves (a network partition can be treated as multiple leaves from the point of view of any subgroup resulting from the partition), there will generally be multiple sponsors. The sponsors need to collaborate to compute secret keys and blinded keys in multiple rounds (limited by $O(log_2 n)$) in a way similar to the initial setting up phase. After all sponsors compute and broadcast all blinded keys, every member will compute the root key. Similarly, multiple joins (and merging) can be performed in multiple rounds.

III. Block-Free Tree-based Group Diffie-Hellman (BF-TGDH) key agreement protocol

A. BF-TGDH principle

As pointed out by the authors [8] of TGDH, during the process of TGDH rekeying, all members stop communication (i.e., block) until the new root key is formed. We present a new protocol called Block-Free Tree-based Group Diffie-Hellman key agreement (BF-TGDH) in this section. The BF-TGDH protocol will allow the group members to continue their seamless communication without interruption during rekeying process regardless of join, leave, multiple joins (merging) and multiple leaves (partition).

The basic idea behind BF-TGDH is as follows: (1) there are two kinds of keys: *front-end* key and *back-end* keys. The front-end key can be computed by all group members whereas for back-end keys, each member will have one key he/she can not compute. (2) whenever a member leaves[1], the remaining members will *switch* to the *back-end* key the leaving member does not have *immediately*. (3) the re-computation of all keys is performed in *background*. There are two meanings regarding background here. One is that the computation of keys are performed during the interval between sending out packets and waiting for receiving packets, thus utilizing the idle time of a computer to compute new keys. The other is that the rekeying materials are appended to out-going data packets, thus, reducing communication cost. The idea looks simple but the difficulty comes from determining how to design and implement back-end keys so that back-end keys are both able to exclude members and their computation remains efficient? Figure 2 illustrates the operation of BF-TGDH.

Based on TGDH, we propose the following mechanism: (1) the front-end key is the root key in TDGH and (2) the back-end keys are computed in an identical way to the root key as follows. Suppose the root key is denoted as $RK(a_1 \cdots a_i \cdots a_n)$ where $a_1, \cdots, a_i, \cdots, a_n$ (correspondingly $b_1 = g^{a_1}, \cdots, b_i = g^{a_i}, \cdots, b_n = g^{a_n}$) are private shares (correspondingly disguised public shares) of $M_1, \cdots, M_i, \cdots, M_n$ respectively. Imagine there are n *dummy member*s $D_1, \cdots, D_i, \cdots, D_n$ and corresponding n *dummy private share*s $d_1, \cdots, d_i, \cdots, d_n$. The members can compute a back-end key, called *dummy root key* and denoted as $DRK_i(a_1 \cdots d_i \cdots a_n)$ $(i = 1, \cdots, n)$, in parallel with the computation of the root key $RK(a_1 \cdots a_i \cdots a_n)$. In DRK_i, the private share a_i is replaced by dummy private share d_i. Therefore, DRK_i can be computed by all members except M_i.

As indicated in the previous sections, the typical problem with the Diffie-Hellman key exchange is the *Man-in-the-Middle* attack and the same problem exists in TGDH. One possible solution is to require that the quantities $(a_1, b_1 = g^{a_1})$,$(a_2, b_2 = g^{a_2})$ in the Diffie-Hellman key exchange be made permanent. Once b_1, b_2 are publicly known in a permanent manner, the Man-in-the-Middle attack will become unsuccessful [13].

[1]The join operation is generally much easier and more efficient to perform than the leave operation and is not described here.

Therefore in BF-TGDH, we assume that every member possesses a permanent pair (private share, disguised public share). As a result, BF-TGDH will resist the Man-in-the-Middle attack. We believe that the idea that every individual will have a permanent pair of Diffie-Hellman shares is reasonable and useful. Just like in the RSA system where every individual has a private key and public key (in a permanent manner) which will allow any individual to communicate with any other individual securely whenever both of them want to, the permanent Diffie-Hellman private share and disguised public share will allow any number of individuals to communicate securely whenever they want to. Moreover authentication is more crucial in group communication than in two-party communication. Since ElGamal public cryptosystem is based on the DLP problem and the setting of BF-TGDH is DLP, the BF-TGDH protocol utilizes ElGamal signature for the authentication of senders instead of a separate RSA signature, which is used in TGDH. In the following paragraphs, we describe BF-TGDH in greater detail.

Suppose that every member M_i has a permanent Diffie-Hellman private share a_i and disguised public share b_i. Moreover suppose that there is an off-line Diffie-Hellman *shares generator*. The *shares generator* generates n (the maximum possible number of members in the group) Diffie-Hellman private shares $d_1, \cdots, d_n$ and their corresponding disguised public shares $e_1 = g^{d_1} \ mod \ p, \cdots, e_n = g^{d_n} \ mod \ p$. $e_1, \cdots, e_n$ are made public. These components are called dummy components. $d_1, \cdots, d_n$ are called *dummy private share*s and $e_1, \cdots, e_n$ are called *dummy disguised public share*s. We also imagine that there are n dummy members $D_1, \cdots, D_n$ who possess these dummy components $(d_1, e_1), \cdots, (d_n, e_n)$ respectively. Assume there is a public one-way function POF.

As in TGDH, each leaf node $< l, v >$ is associated with a member M_i. Moreover, $< l, v >$ is imagined to be associated with a dummy member D_i (See Figure 2). As in TGDH, the group members compute the root key $RK(a_1 \cdots a_n)$, where $a_1, \cdots, a_n$ are permanent private shares of group members $M_1, \cdots, M_n$. Moreover, every member M_i can compute dummy root keys $DRK_j(a_1 \cdots d_j \cdots a_n)$, $j = 1, \cdots, i\text{-}1, i+1, \cdots, n$ in parallel with the computation of the root key RK with a little computation cost but no extra communication cost. For secure group communication, all group members pass the root key RK through the public one-way function POF to get the Data Encryption Key (DEK), i.e., $DEK = POF(RK(a_1 \cdots a_n))$ and then encrypt the message with DEK. Moreover, when a member broadcasts a message, it will sign the message using ElGamal signature protocol. All other members can authenticate the sender.

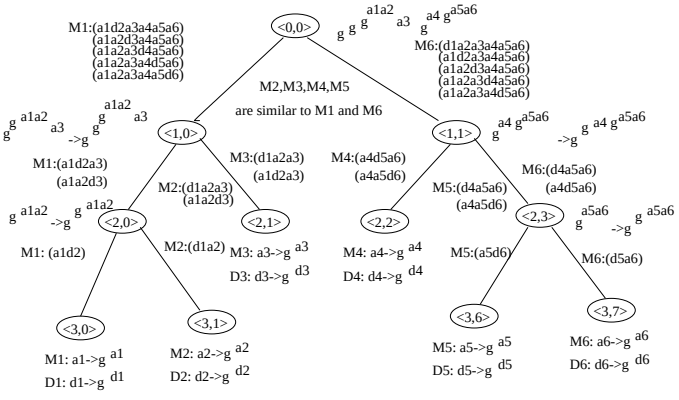

Fig. 2. Block-Free Tree-based Group Diffi e-Hellman key agreement for seamless SGC. The notation $(a_1 \cdots d_i \cdots a_m)$ represents $k(a_1 \cdots d_i \cdots a_m) \rightarrow g^{k(a_1 \cdots d_i \cdots a_m)}$ where $k(a_1 \cdots d_i \cdots a_m)$ is a dummy secret key and $BK(a_1 \cdots d_i \cdots a_m) = g^{k(a_1 \cdots d_i \cdots a_m)}$ is a dummy blinded key. D_i's are dummy members and d_i's ($e_i = g^{d_i}$) are dummy private shares (dummy disguised public shares) which are generated by an off-line shares generator. For example, let us consider M_1. M_1 can compute $(a_1 d_2)$ (i.e., $g^{a_1 d_2} \rightarrow g^{g^{a_1 d_2}}$) and broadcast $BK(a_1 d_2) = g^{g^{a_1 d_2}}$. Then M_1 computes $(a_1 d_2 a_3)$ (i.e., $g^{g^{a_1 d_2 a_3}} \rightarrow g^{g^{g^{a_1 d_2 a_3}}}$ after receiving g^{a_3} from M_3) and $(a_1 a_2 d_3)$ from g^{d_3} which is generated by the off-line shares generator. Finally, M_1 computes $(a_1 d_2 a_3 a_4 a_5 a_6)$ (when receiving $g^{g^{a_4 g^{a_5 a_6}}}$), $(a_1 a_2 d_3 a_4 a_5 a_6)$ (when receiving $g^{g^{a_4 g^{a_5 a_6}}}$), $(a_1 a_2 a_3 d_4 a_5 a_6)$ (when receiving $g^{g^{d_4 g^{a_5 a_6}}}$), $(a_1 a_2 a_3 a_4 d_5 a_6)$ (when receiving $g^{g^{a_4 g^{d_5 a_6}}}$), and $(a_1 a_2 a_3 a_4 a_5 d_6)$ (when receiving $g^{g^{a_4 g^{a_5 d_6}}}$).

B. BF-TGDH rekeying operations

When a member joins the group, the member will broadcast a join request. After receiving the join request, every group member will pass the current DEK through the one-way function to get the new DEK (=$POF(DEK)$), and use the new DEK to encrypt and decrypt the messages without any interruption. The sponsor of the joining member is responsible for sending the new DEK to the joining member. Before sending the new DEK, the sponsor encrypts it with the Diffie-Hellman key between him/her and the joining member. As a result, the joining member can participate in the group communication immediately. During the data communication, the sponsor of the joining member will compute, *in the background*, the secret keys/dummy secret keys and blinded keys/dummy blinded keys on the path from his/her leaf node to the root node, and broadcast the blinded keys/dummy blinded keys to the group by appending the blinded keys/dummy blinded keys to out-going messages or a separate packet if there are no out-going messages. All the group members will compute the new root key and new dummy root keys

after they receive the blinded keys/dummy blinded keys from the sponsor. The sponsor also sends blinded keys and dummy blinded keys, specifically, those keys corresponding to the nodes along the path from the joining member to the root, to the joining member so that the joining member can compute the new root key and new dummy root keys. Once a group member computes the new root key (and new dummy root keys), it computes the new $DEK = POF(RK(a_1 \cdots a_n))$ and uses the new DEK for communication (it is possible for him/her to use the old DEK to decrypt some messages encrypted with the old DEK for a while[2]).

When a member M_i leaves, after receiving the departure request, all remaining members will use the dummy root key DRK_i to get the new DEK (i.e., $DEK = POF(DRK_i)$) and use the new DEK to continue the communication without interruption. However, the leaving member M_i cannot decrypt these messages because its share is not in DRK_i. During the communication, the sponsor of the leaving member will recompute the secret keys/dummy secret keys and blinded keys/dummy blinded keys along the path from him/her to the root and broadcast all blinded keys/dummy blinded keys by appending them to out-going messages.

When multiple members join at the same time, all members will pass the current DEK through the one-way function to get the new DEK and use the new DEK to communicate. There will be multiple sponsors in general and a member may be the sponsor for several joining members. Each sponsor will encrypt the new DEK using the Diffie-Hellman key between him/herself and one of his/her joining members and send the DEK to the joining member. The joining members can participate in the communication immediately. During communication, the new root key and dummy root keys will be reestablished.

When multiple members (say, three) leave at the same time (suppose the members are M_1, M_4, M_6), all remaining members will get the new DEK as follows: multiply the three dummy root keys to get the product $PDRK = DRK_1 \times DRK_4 \times DRK_6$ and then pass the product $PDRK$ through the one-way function to get the new DEK. Since any leaving member will have one dummy root key missing, he/she cannot compute the new DEK. After the new DEK is computed, the group communication continues and the leaving members are excluded. During the process of communication, the sponsors will reestablish the new root key and dummy root keys. Note that if two leaving members collude, they have all dummy root keys for computing the new DEK, so BF-TGDH is not resistant to the collusion of leaving members who leave at the same time. However, this success due to collusion will persist just for a short time. Once the new root key and dummy root keys are reestablished, the leaving members are excluded completely.

When multiple members join and leave at the same time, the new DEK will be the one computed similar to the case of multiple leaves. This new DEK will be sent to joining members immediately. Therefore, the group communication can continue without interruption.

It is worth pointing out that when member(s) join or leave, the (virtual) key tree may need to be adjusted (expanded or compressed) which will be done simultaneously by all members in an identical way.

IV. Discussions

We discuss the performance and security issues of BF-TGDH in this section, specifically, in comparison to TGDH.

The number of rounds for rekeying in BF-TGDH is exactly the same as in TGDH. However all the rounds in BF-TGDH are hidden behind data communication. Therefore there is no interruption, no latency, and more robustness against network congestion and link failure. When computational efficiency is considered, every member will compute n root keys (one root key and n-1 dummy root keys) instead of one root key. It seems that the computational efficiency reduces a lot. However for network based group communication, the communication efficiency through a network is the main concern rather than computational efficiency within a computer, especially since modern computers have huge storage and very high CPU speed. As for dummy blinded keys, since there are a total of n dummy root keys, every member needs to be responsible for computing and broadcasting only one dummy blinded key at each level. This is a very small amount of extra cost compared to the computation of just one blinded key. As for communication cost, since the keys are broadcast by appending them to the out-going messages, there is very little extra cost for communication. In summary, the performance of BF-TGDH is comparable to that of TGDH, with the added benefit of blocking-free communication.

Regarding the security issues, we have pointed out that when two leaving members collude, they can compute the product $PDRK$ of dummy root keys, thus getting the new DEK. However, the success for their collusion lasts just for a short time. Once the new root key is formed the leaving member(s) are excluded completely. One problem with BF-TGDH is its loss of the *perfect forward*

[2]The version number of the DEK used in the encryption may be included in the messages.

security (PFS) because of the permanent property of Diffie-Hellman shares. It can be argued that PFS may not be a requirement for some group communication applications. In case PFS is required, the protocol could use temporary Diffie-Hellman shares, instead of permanent values, which, however, similar to TGDH, will be vulnerable to the Man-in-the-Middle attack. So there is a trade-off between PFS and the Man-in-the-Middle attack. Another solution to thwart the Man-in-the-Middle attack is to use public authentication when exchanging Diffie-Hellman disguised public shares [13].

Apart from the theoretical issues discussed above, there are some practical considerations which include, but are not limited to (1) how serious is the blocking under current SGC protocols over current network environments? (2) how frequently do members join/leave and what are the blocking delays for some typical SGC applications? (3) what are the performances for different SGC protocols? (4) what is the convergence time (i.e., when do all group members obtain the same new keys)? We are performing simulation experiments and testing the performance for BF-TGDH and other protocols to try to answer the above questions.

V. Conclusion

We proposed a Block-Free Group Diffie-Hellman distributed contributory key agreement protocol for seamless secure group communication with the capabilities of preventing the Man-in-the-Middle attack and authenticating senders. The protocol is efficient, robust and easy to implement. Implementing the API for BF-TGDH and testing its performance are our on-going work.

References

[1] L. R. Dondeti, S. Mukherjee, and A. Samal, "Disec: a distributed framework for scalable secure many-to-many communication," *In Proceedings of Fifth IEEE Symposium on Computers and Communications (ISCC 2000)*, pp. 693–698, July 2000.

[2] G. Caronni, K. Waldvogel, D. Sun, and B. Plattner, "Efficient security for large and dynamic multicast groups," *Proceedings Seventh IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '98) (Cat. No.98TB100253). Los Alamitos, CA, USA: IEEE*, pp. 376–383, 1998.

[3] G. Noubir, "Multicast security," *European Space Agency, Project: Performance Optimization of Internet Protocol Via Satellite*, Apr. 1998.

[4] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key groups," *SIGCOMM '98, Also University of Texas at Austin, Computer Science Technical report TR 97-23*, pp. 68–79, Dec. 1998.

[5] X. B. Zhang, S. S. Lam, D.-Y. Lee, and Y. R. Yang, "Protocol design for scalable and reliable group rekeying," *Proceedings SPIE Conference on Scalability and Traffic Control in IP Networks*, pp. 87–108, Aug. 2001.

[6] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," *Advances in Cryptology - EUROCRYPT'94, LNCS, Springer, Berlin*, vol. 950, pp. 275–286, May 1995.

[7] I. Ingemarsson, D. Tang, and C. Wong, "A conference key distribution system," *IEEE Transactions on Information Theory*, vol. 28, no. 5, pp. 714–720, Sept. 1982.

[8] Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," *In Proceedings of the 7th ACM Conference on Computer and Communications Security (ACM CCS 2000)*, pp. 235–244, Nov. 2000.

[9] Y. Kim, A. Perrig, and G. Tsudik, "Communication-efficient group key agreement," *In Information System Security, Proceedings of the 17th International Information Security Conference IFIP SEC'01*, pp. 229–244, June 2001.

[10] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener, "A secure audio teleconference system," *Advances in Cryptology-CRYPTO'88, LNCS, Springer-Verlag*, vol. 403, pp. 520–528, Aug. 1990.

[11] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-hellman key distribution extended to group communication," *ACM Conference on Computer and Communications Security (ACM CCS 1996), New Delhi, India*, pp. 31–37, Mar. 1996.

[12] B. Sun, W. Trappe, Y. Sun, and K. J. R. Liu, "A time-efficient contributory key agreement scheme for secure group communications," *Proceedings of IEEE International Conference on Communications*, vol. 2, pp. 1159–1163, Apr. 2002.

[13] C. Kaufman, R. Perlman, and M. Speciner, *Network security: private communication in a public world*, Prentice Hall, Upper Saddle River, NJ, 2002.