EFFICIENT SCALABLE KEY AGREEMENT PROTOCOLS FOR SECURE[1] MULTICAST COMMUNICATION IN MANETs

Maria Striki and John S. Baras

Electrical and Computer Engineering Department
and the Institute for Systems Research
University of Maryland College Park
College Park, MD 20742

ABSTRACT

In this paper protocols for group key distribution are compared and evaluated from the point of view of Mobile Ad Hoc Networks (MANETs). A MANET is a collection of wireless mobile nodes, communicating among themselves over possibly multi-hop paths, without the help of any infrastructure such as base stations or access points. Thus there is need to render those networks as autonomous and secure as possible, since no central authorization can be assumed at all times. Key management is the service that ensures the security of communication among nodes and the capability of their cooperation as a secure group. It consists of three important services: key generation, user authentication and key distribution. In this work we assume that the participating users have already been authenticated and we are focused only on studying and comparing protocols for group key establishment in MANETs. We distinguish the protocols in two families, contributory and non-contributory and we evaluate them from the point of view of MANETs.

INTRODUCTION

As the development of wireless multicast services such as cable TV, secure audio and conferencing, visual broadcasts, military command and control grows, the research on security for wireless multicasting becomes increasingly important. Information should be communicated to the appropriate groups of nodes with the utmost security and with respect to the network constraints at the same time. It is essential to develop a secure, robust key distribution scheme for multicast communications. Key management determines the security, scalability and efficiency of the network. In this work, we focus on studying and developing key distribution techniques in networks of limited bandwidth, unreliable channels, where topology is changing fast. Nodes within the network may have limited capacity, computational and transmission power (satellites, laptops, PDAs, cell-phones).

Mobile ad-hoc networks are dynamic and connections among nodes are temporary and unreliable. A node may loose its connection to another node(s) because it might either move out of reach or run out of battery resources, or can be compromised. We cannot always assume that a node within the group that has direct connections to all other participant nodes exists. A broadcast from a node to its nearby neighbors only, seems more viable. Also, a change in the topology of a group might occur while the group key is being calculated. In some protocols this event may cause enormous overhead, as the operation of calculating the group key must start over. These constraints render most of the group key distribution protocols inefficient when quick operations to catch up with the rapidly changing topology of the network are required. We distinguish the existing protocols in two categories: *contributory* protocols where all participants take equally part in the key generation and guarantee for their part that the resulting key is fresh and *non-contributory*.

Our objectives in this paper are to study the properties of these two families of protocols, discuss their pros and cons from the perspective of MANETs and evaluate the performance of protocols representing each family. We selected the One-Way Function Tree (OFT) protocol from the family of non-contributory ones, and the 2^d-Octopus protocol (O), with our own modified versions of both **(MO)** and **(MOT)**, to represent the family of contributory protocols. We derive the cost functions for each protocol in terms of total communication, computation and storage of nodes within the group. We also describe in detail how the join and leave procedures occur, both for our protocols and for the original Octopus as well, since this issue is not addressed in the original paper on the Octopus protocol.

(1) The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, neither expressed or implied, of the Army Research Laboratory or the U.S. Government

PREVIOUS WORK

Becker et al. [1], derived lower bounds for contributory key distribution systems for the gossip problem and proved them realistic for Diffie-Hellman (DH) based protocols. Steiner et al.[2], extended the DH protocols to groups. From their work, GDH.2 acquires the alluring property of minimization of total exchanges. TGDH by Kim et al. [10], is a new hybrid, efficient protocol that blends binary key trees with DH key exchange. Becker in [1], introduced the Hypercube protocol as one requiring the minimum number of rounds. In [5], Asokan et al. added ways to recover from node failures. Becker introduced the Octopus protocol that required minimum number of messages and then derived the 2^d-Octopus that combined Octopus with Hypercube to a very efficient protocol that worked for arbitrary number of nodes. Non-contributory protocols are usually based on a simple key distribution center. The simplest is Group Key Management Protocol (GKMP) [9], in which a group leader shares a secret key with each member. The Logical Tree Hierarchy protocol (LKH) [8], creates a hierarchy of keys for each group member. Evolution of the latter is the very efficient OFT [7] that minimizes the number of bits broadcast after a membership change. We selected OFT to represent the family of non-contributory protocols.

SECURITY PROPERTIES AND CONSTRAINTS IN MANETs

The following very important properties have already been proven for the protocols we are discussing.
Forward (Backward) Secrecy: a passive adversary who knows a contiguous subset of group keys cannot discover subsequent (preceding) group keys.
Group Key Secrecy: it is computationally infeasible for a passive adversary to discover any group key.

In the rest of this section we discuss the extra security constraints imposed by MANETs. In [10], blending binary key trees with DH key exchange, results in a secure, simple protocol, TGDH, with no fixed leader for the group. Any trusted member should be ready to become "sponsor" and assume the duties of a leader for a member join/leave operation. If this idea can be extended to MANETs, then no single point of failure – which is a weak point for most non-contributory protocols – will exist. However, any trusted, robust node with certain computational, storage and communication capabilities could become leader, in the same sense that there is a specific node in contributory protocols such as GDH.2 e.g., responsible for the broadcast in round *n*. Even in contributory protocols there exist nodes with some extra responsibilities. The most important *constraint* for MANETs is the existence of robust nodes (e.g. nodes that exit the group less frequently than the rest), capable to become leaders. In a wireless ad-hoc network this is not always possible due to node limitations or network limitations. Thus, protocols like TGDH work under the assumption that the network is relatively small and all nodes acquire the bandwidth resources to broadcast a message that reaches all group members.

In a non-contributory protocol, if a node during group key establishment looses connection (e.g. out of reach, out of battery resources) after having declared interest in participating to the group, then as soon as its absence is discovered, extra time could be given hoping for the node to recover or re-appear. All other nodes proceed normally to the derivation of the group key in the mean time. If the node is still down, then a new group key is computed that excludes the particular node, by updating a restricted number of keys in most cases. Members' contributions may not follow a strict time ordering. However in a contributory protocol it is common for each member to contribute its portion of the key during a fixed slot. If it does not respond during the given slot, the whole procedure comes to a standstill as all further actions of members depend on the contribution of the lost one. In [10] however, authors describe how "cascaded events" are handled (a membership change occurs while another is being handled) for TGDH that is considered contributory. Also, the mechanism invented to make the DH key exchange on a d-cube (contributory) fault tolerant in [5], is less simple than the mechanism inherent in non-contributory tree structures. Contributory protocols are preferred when no previously agreed common secrets exist and parties do not trust each other. But even then, a leader may still be elected. Key agreement is always important since it reflects the totally distributed nature of a group, and no node constitutes a single point of failure.

We have just described the limitations for each family of protocols, and the framework that makes each one efficient. In a MANET, where no trusted third parties are assumed, if we select a non-contributory protocol for key distribution we have to consider the cost for selecting a node as the group leader and for establishing secure initial keys between this node and each member of the group.

SECURE GROUP KEY AGREEMENTS AND OUR EXTENTIONS

We omit the description of **GDH.2**, **OFT**, **TGDH**, and **2^d-Octopus** protocols since they are documented in our references and Technical Report [6].

A. Evaluation of the Cost for the GDH.2 protocol

Initial communication: It takes n rounds (n-1 messages from member M_i to member M_{i+1} of length (i+1) K bits, one multicast message of length nK bits from M_n to the rest n-1 members). The total message length in average is:

$$\sum_{i=1}^{n-1} (i+1)K + nK = (n^2 + 3n - 2)K/2$$

Member initial computation: Member M_i does (i+1) exp/s of the same complexity each. Member M_n does n exp/s. In average, a member does n/2 exp/s and one more when it gets the stream message for the group key.

GSC addition computation: Member M_n generates a new exponent and sends an up-flow message to member M_{n+1} that computes the new key and does n+1 exp/s (n+1 intermediate values).

Add/Delete communication: For the addition case, member M_n sends an up-flow message of length (n+1)K bits to M_{n+1}. M_{n+1} broadcasts to all n members its n+1 intermediate values. The total communication cost is 2(n+1)K bits. For the deletion case, member M_n broadcasts a message of (n-1)K bits.

B. Evaluation of the Cost for the TGDH protocol

TGDH resembles OFT. The basic *differences* are the following: any member of the tree can act as a leader depending on its position in the tree, a member knows all blinded keys of the tree in addition to the secret keys in its path from the leaf to the root, and the merging function is the two node DH key exchange. The secret key x of an internal node s is the result of the DH key exchange between offspring left(s) and right(s) with associated secret keys y and z. Then, $x = \alpha^{yz}$, α^x is the blinded key of node s. We assume that any member at any time can become group leader and broadcast messages to all the members of its group. However, the mobility of nodes might make it impossible for a simple node to broadcast to the whole group. Therefore, this protocol is weak in MANETs, unless the network is rather restricted and nodes stay close to each other throughout the entire session. During the TGDH initial construction, every member becomes a sponsor: it computes and broadcasts to the group the keys of all nodes from the leaf to the root. For every successive level of nodes in the tree, the number of sponsors is reduced to half.

Member/Sponsor Computation/Communication: The sponsor for a particular node s in its path, broadcasts the blinded secret key of s to all members, computes the secret key of the parent node of s and blinds it. In total, 2n exp/s and 2n broadcasts are carried out from all sponsors.

Add sponsor computation/communication: Sponsor generates a new and an intermediate node, gets the blinded new member's key and raises it to the power of its own key. The resulting key becomes the intermediate node's secret key, which the sponsor blinds. Similarly are calculated all updated keys in the sponsor's path from the leaf to the root. So, sponsor does 2h exp/s, and sends to all members the updated h blinded keys. The new member however gets all n blinded keys.

Add/Delete member computation: The new member does h exp/s (using the blinded keys of its co-path) to get the group key. One to (h-1) exp/s are done by the rest of members to compute the group key, since not all blinded keys change for them. In avg. each member does 2 exp/s.

Delete Sponsor Computation/Communication: Sponsor becomes the right-most leaf node of the sub-tree routed at the leaving member's sibling node, which is promoted to replace the leaving member's parent node. The rest is as the add case, so the sponsor does 2h exp/s.

DESCRIPTION OF (MO) AND (MOT) PROTOCOLS

We modified the original Octopus protocol, denoted as **(O)**, by replacing its first step with GDH.2 or TGDH. The protocols derived are denoted as **MO** and **MOT** respectively. The members of the group are divided into 2^d subgroups of equivalent size. The sponsor of TGDH for MOT, and the M_n member of GDH.2 for MO, becomes the sub-group leader (GSC). The papers on (O) do not refer to the case of member addition/deletion. *We analyze all cases and calculate the cost values for (O), MO, and MOT.*

Step1: Each subgroup establishes its own sub-group key, or handles member additions/evictions exactly as indicated by GDH.2 and TGDH protocols.

Step2*:* Identical for all three protocols. For the initial derivation of the group key the 2^d GSCs execute the 2^d–Cube protocol via DH exchanges with initial values the keys they have obtained from step1. It takes d rounds, in each of which we have an exchange of 2^d messages (2 for every application of DH, 2^{d-1} DH pairs/round) for the group key to compute. So, each GSC does 2d exp/s. The total message length is $2^d dK$ bits. The key observation in the case of member addition/eviction is to initially re-compute the subgroup key only for this subgroup in which the change of membership has occurred. We execute step1 only for this subgroup. However, we observe that in step2 not all calculations have to be done anew. The two-party DH exchanges between GSCs whose sub-group keys were not modified at step1 or during the previous rounds of step2 need not be executed again. In the first round 2 GSCs are modified, in the second round 4, in round i, 2^i DH key exchanges are done. In total $2(2^d-1)$ messages are communicated. The GSC of the modified subgroup and its first-round mate participate in all rounds. A GSC participates in $\lceil \frac{d+1}{2} \rceil$ rounds and does $2\lceil \frac{d+1}{2} \rceil$ exp/s in avg.

Step3: Differs for each of the three protocols.

Initial Case: In the example for the simple (O) the authors have used d=2. For d>2, the 3d step should be modified as follows: each of the 2^d participants (GSCs) broadcasts d values to its group. For example if d=3 (GSCs noted as: *A, B, C, D, E, F, G, H*), the following operations occur:

Keys derived after 1st round: a^{AB}, a^{CD}, a^{EF}, a^{GH}.

Keys derived after 2nd round: $a^{a^{AB}a^{CD}}$, $a^{a^{EF}a^{GH}}$

Key derived after 3d round: $a^{a^{a^{AB}a^{CD}}a^{a^{EF}a^{GH}}}$.

GSC *A* communicates the following d parts of the group key to its member j: $a^{a^{a^{EF}a^{GH}}}$, $a^{a^{CD}}$, a^{AB/K_j} for (O), or a^B for MO, MOT. The latter value differs for every member in (O), but is the same for MO or MOT since all members of a sub-group share a common sub-group key. The rest (d-1) values are the same for all members in the subgroup for all three protocols. The GSC broadcast to its members such d values as separate messages and members must do d exp/s to compute the final group key. The aim is that intended recipients only should be able to reconstruct the group key. Each member gets d parts of the key. It raises the first to the power of its own secret or group key and gets the first outcome, then raises the second to the first outcome, gets the second outcome etc.

Addition/Eviction Case: This time (d-1) of the d values need not be broadcast anew since they remain unchanged, and they are already stored in every member from the previous time. So, the GSC sends only one value to every member of its sub-group, the value each particular member requires in order to reconstruct the group key. The GSC in the group of which a change of membership occurred needs to communicate to its members at the 3d step no parts for MO and MOT, and one part only for (O). The sub-group has calculated its updated subgroup key already, and its members can use the (d-1) values they have stored to reconstruct the sub-group key. We illustrate the case by continuing with the example where d=3. Assume that a change of membership occurs in the sub-group of A and its new subgroup key is x. GSC *A* stores the following values after three rounds: $x, a^B, a^{a^{CD}}$, $a^{a^{a^{EF}a^{GH}}}$, $a^{a^{a^{AB}a^{CD}}a^{a^{EF}a^{GH}}}$. The intermediate values of *A* remain unchanged from the previous time and will be used by the members of its subgroup to construct the group key. The broadcast values are blinded, so their free communication over the network causes no harm to the security of the system. Also, not all members do d exp/s as in the initial case: members of 2^{d-1} GSCs do one, members of 2^{d-2} GSCs do two, and members of the last two GSCs (*A, B* in the example) do d exp/s to reconstruct the new group key. The number of messages communicated at step3 for MO and MOT is $2^d d$ for the initial and (2^d-1) for the addition/eviction case.

Important Security Constraint:

(O): In the case of member addition/eviction we do not acquire the GDH.2 or the TGDH properties to disguise the contributions of the rest of the members of the subgroup to the new/evicted member. If the protocol is used as such, then there is not backward secrecy. When the GSC sends the appropriate d values to the new member it essentially sends to it the old group key. Obviously, there is no forward secrecy for the eviction case because the new group key is no other than the previous, without the evictee's contribution, which is of course already known to the evictee. So, we do the same modification as in GDH.2 and other secure protocols: another member of the same subgroup modifies its secret share and establishes a new DH key with its GSC. So, the GSC in the case of join (evict) does two (one) DH key exchanges.

(MO), (MOT): At step1, GDH.2 and TGDH produce subgroup keys: $a^{ab\cdots z} = a^N$ and $a^{xy} = a^N$ respectively. These keys are equivalent to the subgroup key produced by (O) when its subgroup contains one member only. A designated member or sponsor from each subgroup becomes GSC and provides the subgroup key for step2. All 2^d GSCs broadcast to all members of their subgroup *the same* parts of key. Operations to derive the group key are exactly the same for these members.

.

SECURITY ISSUES OF (MO) - (MOT) PROTOCOLS

We argue that the security of the new protocols is at the same levels as the security of the original Octopus. In step1 MO and MOT inherit the properties of GDH.2 and TGDH secure protocols respectively. Step2 and step3 serve in calculating the group key for all the participants and distributing it to all the members according to the principles of DH key exchange, so the protocols inherit the security DH key exchange provides for the latter steps.

COMPARISON

Cost	2d-Oct. (O)	Mod. 2d-Oct. (MO)	Mod. 2d-Oct. (MOT)
Initial GSC comput.	$(2\lceil (n-2^d)/2^d \rceil + 2d + \lceil (n-2^d)/2^d \rceil) C_E + (\lceil (n-2^d)/2^d \rceil)^{4/3} + 1.25(\lceil (n-2^d)/2^d \rceil) K^2$	$(\lceil n/2^d \rceil + 2\ d)\ C_E$	$(2\log \lceil n/2^d \rceil + 2\ d)\ C_E$
Initial members comput.	$(d+2)C_E$	$C_E((1/2)\lceil n/2^d \rceil + d)$	$C_E\ (2\log \lceil n/2^d \rceil + d)$
Initial Commun.	$(2n+(d-1)\ 2^{d+1})\ K$	$(2^{d-1}(\lceil n/2^d \rceil^2 + 3\lceil n/2^d \rceil - 2) + 2^d d + 2^{d+1})K$	$(2^d 2\lceil n/2^d \rceil + 2^{d+1}\ d)\ K$

		$+2^d d+2^{d+1})K$	
Add GSC comput.	$(3\lceil (n-2^d)/2^d \rceil+4+2\lceil \frac{d+1}{2} \rceil) C_E +2(\lceil (n-2^d)/2^d \rceil-1) K^2$, one $(2\lceil (n-2^d)/2^d \rceil+2\lceil \frac{d+1}{2} \rceil)C_E$ rest	$C_E(\lceil n/2^d \rceil+1+2\lceil \frac{d+1}{2} \rceil)$, one $C_E(2\lceil \frac{d+1}{2} \rceil)$, rest	$C_E(2\log \lfloor \frac{n+1}{2^d} \rfloor+2\lceil \frac{d+1}{2} \rceil)$, one $2C_E\lceil \frac{d+1}{2} \rceil$ rest
Add members comput.	$4C_E$, two $2C_E$, the rest	$3C_E$, one GSC $2C_E$, rest	$4C_E$, one GSC $2C_E$, rest
Delete Commun.	$(2\lceil (n-2^d)/2^d \rceil+3 2^d-2) K$	$(\lfloor \frac{n-1}{2^d} \rfloor+3 2^d-2)K$	$(\log \lfloor \frac{n-1}{2^d} \rfloor+3 (2^d-1))K$

Table 1: Complexities of key agreement protocols

Table1 shows some of the comparison results. The performance evaluation for OFT can be found in [6], [7]. In terms of Initial GSC Computation, MOT behaves better than all the other protocols. In terms of GSC Add/Evict Computation, (O) is the worst, MO and mainly MOT performs much better when *d* is small. As for the Initial Communication, MO is the worst, MOT and (O) however slightly outperform OFT. For the Addition/Eviction Communication (very critical issue for MANETs), (O) is outperformed by both MO and MOT, and MOT gets closer to OFT than any other contributory protocol.

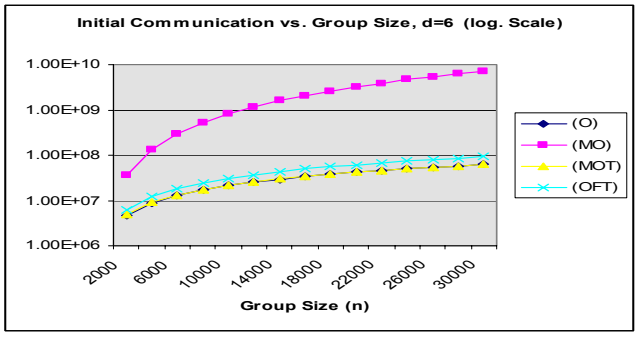

Figure1: Initial Communication vs. Group Size, d=6. MOT and (O) achieve almost the same lowest cost, OFT is slightly worse.

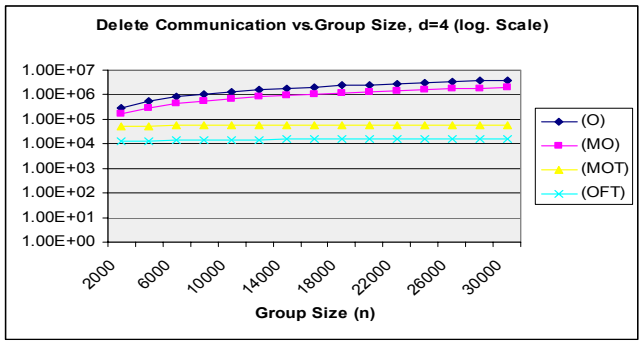

Figure2: Delete Communication vs. Group Size, d=4. OFT achieves the lowest overhead. MOT gets very close to OFT, but the overheads of MO and (O) are still much worse.

SUMMARY AND CONCLUSIONS

The paper discusses the framework and constraints under which already existing protocols can become scalable and robust in MANETs. It distinguishes protocols in two families (contributory/non-contributory), discusses their limitations and suggests solutions to make them applicable in MANETs. We present two novel hybrid protocols MO and MOT-based on the original Octopus (O)-, developed as an attempt to render contributory protocols scalable and efficient. All three are described in detail and cost functions in terms of communication and computation are derived for all operations. From our performance evaluation, we see that MOT outperforms (O) in all cases, and OFT in the computation costs. As for Communication Addition/Eviction costs, MOT gets very close to OFT.

REFERENCES

[1] Klaus Becker, Uta Wille. Communication Complexity of Group Key Distribution. Proc.5th ACM Conference on Computer & Communications Security, pages 1-6, San Francisco, CA, November 1998. ACM Press.

[2] Steiner M., Tsudik G., Waidner M., Diffie-Hellman. Key Distribution Extended to Groups. 3rd ACM Conference on Computer & Communication Security, ACM Press, 1996.31-37

[3] Maarit Hietalachti. Key Establishment in Ad-Hoc Networks. Helsinki University of Technology. Laboratory for Theoretical Computer Science.May'01.

[4] A.Perrig. Efficient Collaborative Key Management Protocols for Secure Autonomous Group Communication. Int'l Workshop on Cryptographic Techniques E-Commerce CryptTEC'99.

[5] N.Asokan and Philip Ginzboorg. Key-Agreement in Ad-Hoc Networks. Elsevier Preprint.2000.

[6] Maria Striki, John S. Baras. Efficient Scalable Key Agreement Protocols for Secure Multicast Comm/tion in MANETs. CSHCN Technical Report 2002.

[7] David McGrew. Alan T.Sherman. Key-Establishment in Large Dynamic Groups Using One-Way Function Trees. May 1998.

[8] H. Harney, E.Harder. Logical Tree Hierarchy protocol. Internet Draft, Internet Eng. Task Force, April 1999.

[9] H. Harney, C.Muckenhirn. Group Key Management Protocol (GKMP). Specification/Architecture, Internet Eng. Task Force. July 1997.

[10] Y. Kim, A. Perrig, G. Tsudik Simple & Fault Tolerant Key Agreement for Dynamic Collaborative Groups.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, neither expressed or implied, of the Army Research Laboratory or the U.S. Government