KEY DISTRIBUTION PROTOCOLS FOR SECURE MULTICAST COMMUNICATION SURVIVABLE 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 we develop a secure, robust and scalable key management scheme for multicast communications in Mobile Ad Hoc Networks (MANETs). Most key distribution protocols of today are primarily designed for wire-line networks. Frequent node failures, network partitions, inefficient computational and communication capabilities of wireless nodes, network delay, bad quality of signal etc, are some of the reasons why they fail to work properly in MANETs. We classify existing and newly developed protocols in two families, contributory and non-contributory. We evaluate them and compare their performance. We focus on studying and developing key distribution techniques that achieve scalability and high performance of our framework without sacrificing the security level of the network. To this end, we also designed a hierarchical two level hybrid key management scheme that utilizes some of the above protocols in the appropriate combinations to further reduce the storage, communication and computation costs of nodes.

INTRODUCTION

A MANET is a collection of wireless mobile nodes, possibly heterogeneous, communicating among themselves over possibly multi-hop paths, without the help of fixed infrastructure. In MANETs, high mobility may result in nodes frequently going out of range or running out of battery, leading in temporary links. Collisions, low link quality, distance between nodes and various other factors result in unreliable links or excessive delay in the network. Existence of leader nodes with direct connections to all other members, able to broadcast to the whole group may not be always guaranteed. The computational power of nodes is considered an issue for some wireless mobile nodes due to resources or capacity limitations. Changes in the topology of a group might occur during the calculation of the group key. Some protocols handle this situation very inefficiently by starting over the whole operation. Such constraints render most key distribution protocols inefficient in environments that require fast operations with low overhead. Due to the increasing demand for secure and scalable multicast services - may be as diverse as cable TV, secure audio, conferencing, visual broadcasts, military command/control - of today, designing secure, efficient and scalable key distribution protocols for multicast communications becomes increasingly important. Specifically, protocols that are robust enough to survive or tolerate frequent node failures, network partitions and merges, delays in critical messages, ambiguity to determine the state of group members under certain circumstances, extensive computations etc., are needed. In this work we assume that the participating members are already authenticated and we focus on the issue of key distribution only.

(*) Research partially supported by the U.S. Army Research Laboratory under Coop. Agreement DAAD19-01-2-0011.

Since MANETs are deployed in diverse environments, the requirements and constraints for nodes that operate in these environments may substantially vary. Consequently, for the design of appropriate key distribution protocols for MANETs, all parameters that distinguish one environment from another must be taken into account. To this end, we distinguish MANETs for key distribution in:

-**Over-layered oriented**: Sufficient trusted-entities, special nodes or hubs exist, accessible from all nodes in the network.
-**Flat oriented**: Few, if any trusted special hubs or nodes exist, that may not be accessible to all nodes at all times, high-level of self-organization is thus expected.
-**Military oriented**: Heterogeneous environment and nodes, assumed to be a combination of the above frameworks.

In this paper we are primarily interested in *Flat* and *Military* oriented MANETs, and for these frameworks we consider the following aspects of *Fault Tolerance* for key distribution protocols: a single, non-flexible, "omnipotent" group leader being a single point of failure, protocols able to recover from members' failure during the group key establishment, and protocols that tolerate frequent node failures, group partitions and merges at any time during a session. In this perspective, we attempt to determine the properties key distribution protocols should acquire to operate properly in each of these frameworks, and then design the appropriate protocols tailored for the requirements of the above frameworks. To this end, we classify existing protocols in two families: *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*, where group key generation does not require equal participation from all members.

Most **non-contributory** protocols are based on a fixed trusted leader to distribute the key. Due to factors of high mobility, poor resources and low link quality encountered in MANETs, group leaders frequently become faulty or unavailable to certain group

members. Finding members within the group able to replace a faulty leader is not enough. In addition, the new leader should securely and quickly obtain information gathered by the previous leader up to that point. It would be preferably selected among group members (*as in contributory protocols*) and have a rather coordinating role, storing as minimal information as possible that can be easily retrieved by any future leader (*e.g. TGDH*). Furthermore, in order to reduce group partitions and frequent leader elections, mobility of nodes within the network and processing capabilities of individual nodes should be considered. Therefore, in a Flat Oriented MANET that demands a high level of self-organization we require that the operation of leader election be dynamic and flexible, according to some intelligent policy for MANETs (e.g. select the node that stays connected with the largest number of nodes within its group for the largest amount of time). The leader should also operate in a rather restricted area of the network. In most **non-contributory** protocols (tree-based), in the event of a node failure, a new group key is computed by updating only a restricted number of keys. The contributions of members for the key establishment are independent and need not follow a strict ordering. In the event of a node failure or delay to respond, the rest of the nodes proceed normally to the key establishment process.

In a **contributory** protocol like GDH.2, each member is expected to contribute its portion of the key according to strict ordering. If a node does not respond during the given slot, the whole procedure comes to a standstill since all further actions of members depend on the contribution of the "faulty" member. Since it may be impossible to determine on time if the response of the node is simply delayed or lost, or if the node itself is down or out of reach, inevitably the key establishment process starts all over again. Nevertheless, these protocols still acquire some very important properties: they are most appropriate when no previous trust is established among nodes, they reflect the distributed nature of a group, and their nature is such that no node constitutes a single point of failure.

The environment of Over-Layered MANET is the least complex to consider. It assumes sufficient trusted entities with special capabilities that rarely become faulty and are accessible to all nodes at any time. Thus, non-contributory centralized protocols that are efficient in terms of performance and scalable as we have just seen, can be supported. The existence of a single, non-flexible group leader is not an issue for this environment. In the Flat Oriented MANET however, it is clear that non-contributory protocols as such cannot be applied to a large area of the network at least. It would be desirable to derive an efficient fault-tolerant hybrid protocol (probably contributory) that combines the main advantages of the two families of protocols.

We claim that the contributory 2^d-Octopus protocol (O) that is based on the Hypercube key exchange scheme is quite appropriate for *Flat Oriented MANETs*: it tolerates various kinds of failures or resumes from failures with minimal overhead. Focused on this environment, we developed two *novel* hybrid protocols based on (O) - GDH.2-based (MO) and TGDH-based (MOT) - that are more efficient in terms of Computational and Communication Cost. The fault tolerance of Octopus protocols has been demonstrated by examining scenarios of failures most likely to occur in MANETs [6]. We gain insight about the extra overhead required to render key distribution protocols scalable and applicable in Flat MANETs by additionally comparing those protocols to the very efficient non-contributory OFT [6].

For the *Military Oriented MANET* we assume diverse circumstances prevailing in different parts of the network. We intend to link key distribution schemes to network topology, hierarchy, predicted or unpredicted member mobility, routing. Heterogeneous nodes (vary from Satellites, UAVs, PDAs, laptops, to GPS devices, cell-phones and pagers) cause links of variable qualities. The communication paths may be uni-directional or bi-directional and asymmetric. We expect larger bandwidth resources at higher tiers (satellites, UAVs), and restricted resources at lower tiers since the devices utilized are much less powerful. Again, the physical and communicational mobility levels at higher and lower tiers are much different. At the low end mobility is more rapidly changing, and higher degree of self-organization is observed. Nodes may need intermittent connectivity to reach others at the higher end. At the high end, it can be assumed that there exist trusted powerful nodes able to broadcast to the whole group(s), become group leaders, or trusted key distribution centers or both. Taking the diversity of this topology into account, we designed a novel *Two-Level Hierarchical Hybrid Key Management Scheme* for multicast group communications that exploits the diversity of the battlefield the best possible way. Our key distribution scheme takes these environmental variations into account and models them so that the first level of the scheme represents nodes at the higher end, and the second level nodes at the lower. We claim that our scheme is most suitable for the "*Military*" MANET and we prove this with our analytical results and calculations.

PREVIOUS WORK

Becker and Wille [1] derived lower bounds for contributory key distribution systems from the results of the gossip problem and applied them to DH-based protocols. They used the basic DH distribution extended to groups from the work of Steiner *et al* [2]. TGDH by Kim *et al* [10], is a new hybrid, efficient protocol that blends binary key trees with DH key exchange. Becker *et al* [1], introduced the Hypercube protocol as one requiring the minimum number of rounds. In [5], Asokan added to the Hypercube protocol 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 an arbitrary number of nodes. We recently developed Modified Octopus Protocols (MO) and (MOT) that are scalable, fault-tolerant and succeed in reducing the communication and computational overhead of original 2^d-Octopus [6].

Most protocols from the non-contributory family are based on a simple key distribution center. The simplest is Group Key Management Protocol (GKMP) [9]. The Logical Tree Hierarchy (LKH) [8], creates a hierarchy of keys. It is more complicated but more efficient. Evolution of the latter is OFT [7]; it minimizes the number of bits broadcast to members after a membership change.

BRIEF INTRODUCTION TO PROTOCOLS USED IN THE TWO-LEVEL HYBRID SCHEME

1. Core Based Tree Protocol (CBT)

It acquires the tree structure. The leaf nodes are the group members (n) and the group key is associated with the root of the tree. Each member knows all keys from its leaf node up to the root, but no other node in the tree.

Member joins group: It is assigned to a leaf node and receives all keys on the key path to the root by the key server. All keys received are independent from any previous keys (backward secrecy). The key server sends each of these new keys to the appropriate members of the group on a "need to know" basis.

Member leaves group: all keys the evicted member knows are changed (forward secrecy). They get replaced sequentially from the leaf up to the root key. The update involves $\log_d(n)$ keys.

2. Group Key Management Protocol (GKMP)

It uses a Single Group Security Agent (GSA) to distribute the appropriate pairs of keys (individual and session key) to the members. Initially the GSA creates the n secret keys it is going send to the n members of the group via the Public Encryption method. Then, it creates a group session key for multicast communication that communicates to each member individually using Symmetric Encryption this time. The session key is encrypted with the private key sent earlier to each member.

3. One-Way Function Tree (OFT)

It is an efficient evolution of the CBT scheme. Interior nodes of the tree have exactly two leaves. Node x is associated with a secret key k_x and a blinded key $k_x' = g(k_x)$, (computationally limited adversary may know k_x' and yet cannot find k_x); g is a one-way function. Each member knows the un-blinded node keys on the path from its node to the root, and the blinded node keys that are siblings to the nodes in its path to the root.

Members compute un-blinded keys along their path. The manager assigns and securely communicates a randomly chosen key to each member. It securely communicates with subsets of group members via symmetric encryption. If a blinded key changes and the member gets the new value (sent to all members that store it), it re-computes the keys on the path and finds the new key. The interior node keys are defined by the rule: $K_x = f(g(k_{left(x)}), g(k_{right(x)}))$, where $left(x)$ and $right(x)$ denote the left and the right children of the node x, and f is a mixing function.

4. Octopus Protocol

Four parties *A, B, C, D* generate a group key by four exchanges only. First, *A* and *B*, then *C* and *D* do a DH key exchange generating keys α^{ab} and α^{cd}. Then, *A* and *C* as well as *B* and *D* do a DH key exchange using as secret values the keys generated in the 1st step. *A(B)* sends $\alpha^{\phi(a^{ab})}$ to *C(D)*, which sends $\alpha^{\phi(a^{cd})}$ to *A(B)* so that *A* and *C* (*B, D*) generate joint key $\alpha^{\phi(a^{cd})\phi(a^{ab})}$.

Parties $P_1, P_2,\ldots, P_{n-4}$, *A, B, C, D* generate a common group key by dividing themselves into five groups. *A, B, C, D* take charge of the central control. The remaining parties distribute themselves into 4 groups: $\{P_i \mid i \in I_A\},\ldots, \{P_i \mid i \in I_D\}$, where I_A, I_B, I_C, I_D are pair-wise disjoint, and $I_A \cup I_B \cup I_C \cup I_D = \{1,...,n-4\}$. $P_1,\ldots,P_n$ generate a group key as follows:

1. $\forall X \in \{A, B, C, D\}, \forall i \in I_X$, *X* generates a joint key k_i with P_i via the DH key exchange.

2. A, B, C, D do the 4-party key exchange using values: $a=K(I_A),\ldots,d=K(I_D)$, where $K(J):=\prod_{i\in J}\phi(k_i)$ for $J\subseteq\{1,..,n-4\}$ and hold the joint key $K= a^{\phi(a^{K(I_A\cup I_B)})\phi(a^{K(I_C\cup I_D)})}$.

3. The step is described only for *A*. Parties *B, C, D* act accordingly. $\forall j \in I_A$, *A* sends 2 values to P_j: $a^{K(I_B\cup I_A\setminus\{j\})}, a^{\phi(a^{K(I_C\cup I_D)})}$. P_j derives $(a^{K(I_B\cup I_A\setminus\{j\})})^{\phi(k_j)} = a^{K(I_A\cup I_B)}$ first, then $K= a^{\phi(a^{K(I_C\cup I_D)})\phi(a^{K(I_A\cup I_B)})}$.

5. Hypercube Protocol

It minimizes the number of simple rounds. 2^d parties agree upon a key within d simple rounds by performing DH key exchanges on the edges of a d-dimensional cube. We identify the 2^d parties on the d-dimensional space $GF(2)^d$ and choose a basis $b_1,\ldots,b_d$ of $GF(2)^d$. In round 1, every participant v generates a random number r_v and does a DH key exchange with participant $v+b_1$ using the values r_v and r_{v+b1}. In round i, every participant v does a DH key exchange with $v+b_i$, where both parties use as secret value the one generated in round i-1. In every round, parties communicate on a maximum number of parallel edges of the d cube. All parties share a common key at the end.

6. 2^d*- Octopus Based Protocol*

For an arbitrary number of participants that require low number of rounds, the idea of Octopus is generalized. In 2^d–Octopus protocol (O), participants act as in original Octopus. However, 2^d instead of four parties are distinguished to take charge of the central control. The remaining n-2^d parties divide into 2^d groups.

Modified Octopus (MO) and (MOT) are based on GDH.2 or TGDH respectively: they maintain the 2nd step of the original 2^d-Octopus intact and substitute the centralized scheme of the 1st and 3d step with GDH.2 or TGDH. The sub-group key becomes: $a^{ab\cdots z} = \alpha^N$ in the case of (MO) and $\alpha^{xy} = \alpha^N$ in the case of (MOT). During the 1st step each sub-group establishes its own sub-group key and handles member additions/evictions exactly as indicated by GDH.2 and TGDH. In both modified protocols the sub-group key acquires the desired structure. However, note that *not all protocols* are appropriate to be incorporated in (O).

7. GDH.2 Protocol

Upflow: each member M_i composes and sends to member M_{i+1}, i intermediate values (each with (i-1) exponents) and one cardinal (i exponents).

Downflow: member M_n is the first to compute the group key. It broadcasts the intermediate values to all members.

8. Tree Group DH (TGDH) protocol

TGDH protocol resembles OFT. The basic differences are the following: any member of the tree can act as a leader or sponsor depending on its position in the tree, a member knows all blinded keys of the tree at any given time, and the merging function is the two-party DH key exchange. The secret key x of an internal node s is the result of the DH key exchange between its offspring $left(s)$ and $right(s)$ with secret keys y and z. Then,

$x=\alpha^{yz}$ and the blinded key of node *s* is α^{x}. During initial tree construction every member becomes sponsor: computes nodes from the leaf to the root and broadcasts them to the group. For every successive node level in the tree, the number of sponsors is reduced to half [6]. Each member knows all keys in its path from the leaf to the root and all blinded keys of the tree.

BRIEF DISCUSSION ON NOVEL MO AND MOT

In [5] the authors claim that the Hypercube Scheme is fault tolerant but they don't analyze all the group disruption cases that may occur in a MANET. In [6] the most frequent scenarios of failure have been examined and we have shown that the protocol can be slightly altered to be made fault tolerant for these cases without significant overhead. There is analytical description of MO and MOT in our Technical Report [6]. The Hypercube Scheme is the core Protocol (2^{nd} step) for all Octopus based ones. All three steps of those protocols are independent modules: *overall Fault-Tolerance* requires Fault-Tolerance for each step individually. Here is where the key distribution scheme of each individual sub-group of Octopus protocols comes into play.

The subgroup key distribution protocols in (O) and (MO) assume either the existence of a single fixed subgroup leader (GKMP) or strict ordering during key establishment and cannot tolerate delays and node failures (GDH.2). Thus (O) and (MO) are not Fault Tolerant overall. TGDH is selected for subgroups in MOT. This protocol assumes that any node should be ready to become sponsor and that the same amount of information is stored in all members with no considerable overhead. Since the subgroup is relatively small in size now and can be deployed on a restricted area of the network also, TGDH can be considered fault-tolerant and applicable for flat MANETs. Under these circumstances, we claim that MOT is scalable and Fault-Tolerant overall.

In Octopus-based protocols, each member of the hypercube is the leader of a subgroup of nodes. In (O) members of a subgroup establish a two-party DH key with the subgroup leader. These partial keys of members are used to construct the initial secret share for the hypercube. After the group key is derived, the 2^d leaders distribute parts of the group key to their sub-group members in a way that a member that does not belong to the sub group cannot derive the group key. The key distribution protocols that these sub-groups support may be selected with much more freedom however. Furthermore, ach of these subgroups is deployed on a relatively *restricted* area of the network and it is easier to handle these groups in a *localized manner*. Given the topology of the network we have the freedom to assign to each of these sub-groups arbitrarily many members. This results in less traffic for the routing protocol and less bandwidth consumption. Moreover, if a GSC becomes "faulty", it can be replaced by another node from its own sub-group. It is clear now how these properties render the protocol robust in the cases of addition/deletion, merging/partition.

Analytical performance evaluation has been conducted for all the protocols mentioned up to now. It can be found in [6], [11]. It is omitted here for luck of space.

TWO LEVEL HYBRID MODEL

The scheme has the following modules: In the upper level the Group Security Controller (GSC) node is the leader of the upper level group built from all the nodes called Group Security Agents (GSAs). Every GSA is the leader of a group of simple members of the second level, and in practice it is dynamically selected. The GSC is also leader of all the group members of the second level. In the upper level we can assume a satellite or a UAV as GSC. It has relatively low mobility but high bandwidth and processing capabilities. It controls the GSAs (n_1) and the members (n_1 n_2), so it is responsible for the whole network. GSA controls one group of members only, so the requirements for energy, bandwidth and computation power can be lower than those of the GSC. Every multicast group acquires its own group key. Members of the same group acquire similar mobility and behavior patterns. The key management is independent for each subgroup and the update is limited to the subgroup. We investigate the effect of mobility and link failure in a MANET by providing corresponding values for the probabilities or mobility frequencies p_1, p_2: $p_1 < p_2$

The most efficient and robust among the key distribution schemes described above are incorporated into the two-level hybrid scheme, in various combinations and we conduct an overall performance evaluation of the model for every such combination. The results demonstrate which combination presents the best overall performance given the ratio of users at each level (n_2/n_1), the relative ratio of the mobility of users (p_2/p_1) and parameters that determine the level of security: length of key (K) etc. This way we obtain a theory and a tool for evaluation and design so that given the parameters of the network we can decide on the most appropriate version of the two-level hybrid model for that particular case.

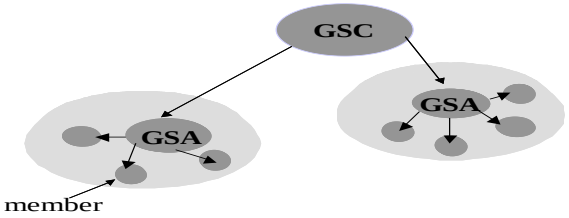

Figure 1: The two-level hybrid key management scheme

PARAMETERS

In order to get closed analytical expressions for the cost functions, we have derived the computation cost for the random function generator **(C_r)**, for Public Key Infrastructure (PKI) encryption/decryption **(C_{PE}/C_{PD}),** for symmetric key encryption/ decryption **(C_{SE}/C_{SD}),** for hash functions, and exponentiations C_E. In [11] we show how these values and all the costs of the scheme as a consequence, depend on the number of users ***n***, key tree parameters of descendants and tree height **(d, h),** the frequencies of the member motion **(p)** and the length of keys **(K).** We select the **RSA** method for modeling the PKI, and the **DES** method for the symmetric encryption/decryption.

C_r	C_{rr}=(Klog2/2) + (K/k) K^{-2} pseudo- random number generation C_r= ((1.4/k)+2.8) K^4 pseudo-random generation for primes (used for RSA keys)
C_{PE}	(4 K_e+2$\log_2 K_e$) K^2
C_{PD}	((4/25)K_d+(2/25)$\log_2 K_d$)K^2
C_{SE}	C_{DES} K
C_{SD}	C_{DES} K
C_E	(4l+2$\log_2$l)K^2
C_g	30K

Table 1: Estimation of Costs of Security Parameters used in the Protocols

SELECTED SCENARIOS

We primarily wish to minimize the avg. overall Comm/tion or avg. GSC, GSA or member Operating Costs for adding or deleting a GSA or member. Assume that addition and deletion occur with equal probability (1/2).

GSC is considered robust and trusted: thus for upper level centralized non-contributory schemes are preferred since they generally present better overall performance. For **GSAs** trust maybe unknown but GSC monitors GSAs. Centralized or contributory schemes may be selected for lower level, depending on network assumptions, topology, group size and mobility.

Our Hierarchical Scheme provides *adaptability* to *network limitations* and *topology*. It may use a combination of protocols to achieve either substantial communication overhead reduction, or lower the computation cost or both. It may support a wide range of protocols at any time. Given the parameters of the network like users mobility frequencies, user characteristics (processing capabilities, bandwidth, battery life), overall number of users and number of users of certain categories, it can select the most appropriate key distribution protocols for both its levels and incorporate them into a very efficient hybrid scheme. The performance evaluation of the individual protocols we conducted in [11] is an auxiliary step for the evaluation of the Two-Level Hybrid Scheme. The most efficient {*GKMP, CBT, ELK, OFT, MOT, GDH.2, TGDH}* have been selected and inserted in the scheme in combinations that make sense:

Single Key Tree, two Groups of Members (scheme1)
GSC to GSAs Key Tree, each cluster GKMP (scheme2)
GSC to GSAs Key Tree, each cluster Key Tree (scheme3)
Single Key Tree, single Group of Members (scheme4)
GSC to GSAs GKMP, each cluster Key Tree (scheme5)
GSC to GSAs GKMP, each cluster GKMP (scheme6)
GSC to GSAs OFT, cluster {OFT, ELK, GDH.2, MOT, TGDH}
GSC to GSAs TGDH, cluster {OFT, GDH.2, MOT, TGDH}
GSC to GSAs ELK, cluster {OFT, ELK, GDH.2, MOT, TGDH}

GRAPHIC RESULTS AND DISCUSSION

The following table shows some comparison results of Octopus-based protocols. MOT outperforms the rest in terms of Initial and Add/Evict Comp/tion. (O) is far the worst, MO is much better for cases of small *d*, large *n*. It behaves poorly in terms of Initial Comm/tion. MOT and (O) perform much better. For Add/Evict Comm/tion, MO and MOT outperform (O). MOT gets closer to OFT than any other contributory protocol.

Cost	2^d-Octopus (O)	Mod. 2^d- Oct (MO)	Mod.2^d-Oct (MOT)
Init GSC Cmp	$(3\lceil n-2^d/2^d \rceil+2d)C_E+ K^2$ $(\lceil n-2^d/2^d \rceil^{4/3}+1.25 \lceil n-2^d/2^d \rceil) K^2+\lceil n/2^d \rceil C_{rr}$	$(\lceil n/2^d \rceil+2d)\ C_E + \lceil n/2^d \rceil\ C_{rr}$	$C_E(2\log \lfloor n/2^d \rfloor+2d)+ \lceil n/2^d \rceil C_{rr}$ max.
Del. GSC Cmp	$C_E(3\lceil n-2^d/2^d \rceil+2+2 \lceil \frac{d+1}{2} \rceil)+2\ (\lceil n-2^d/2^d \rceil -3)\ K^2 + C_{rr}$, one $2(\lceil n-2^d/2^d \rceil+\lceil \frac{d+1}{2} \rceil)C_E$ rest	$C_E(\lfloor n/2^d \rfloor+2 \lceil \frac{d+1}{2} \rceil) +C_{rr}$, one $C_E(2\lceil \frac{d+1}{2} \rceil)$, rest	C_E (2log $\lfloor \frac{n+1}{2^d} \rfloor+2\lceil \frac{d+1}{2} \rceil)+ C_{rr}$, one $2C_E \lceil \frac{d+1}{2} \rceil$ rest
Del Cmm	$(2\lceil n-2^d/2^d \rceil+32^d-2)\ K$	$(\lfloor \frac{n-1}{2^d} \rfloor+3\ 2^d-2)K$	$(\log \lfloor \frac{n-1}{2^d} \rfloor+3(2^d-1))K$

Table 2: Complexities of Octopus based protocols

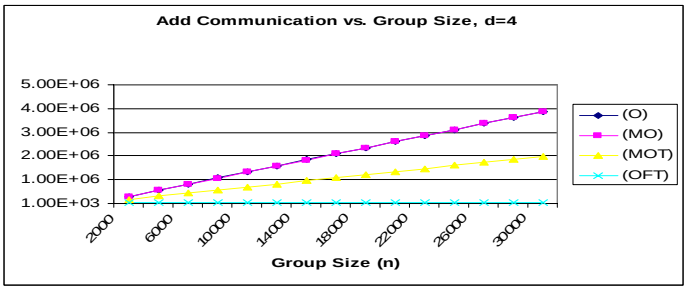

Figure1: Add Comm/tion vs. Group Size, d=4. OFT achieves lowest overhead. MOT gets quite close to OFT, overheads of MO, (O) are similar but much worse.

Two-Level Hierarchical Scheme: For *Initialization* and *Operative* Costs of *GSC* and *Members* the *two levels* of are assumed *independent modules* that do not interact. The resulting Costs are derived from the performance evaluation of the individual protocols applied to the first and the second levels. The *Initialization and* Operative Costs for *Total Communication* and *GSA Computation* are *combination* of both levels. The resulting costs do not stem directly from the individual performance of each of the protocols.

Assume (n_2/n_1) = t(1, 4, 16) and n_1=(32, 16, 8), n_2=(32, 64, 128), (p_2/p_1) = y(2.5, 10 ,40), p_1=(0.08, 0.04, 0.02), p_2 =(0.2, 0.4, 0.8) Metrics *are very sensitive* to t and y (operating costs), to values K and consequently to all key *encryption/decryption costs*. Naturally, metrics *are sensitive* to *p1*, *p2*, *n1*, *n2* alone.

The general performance of the *Two-Level Hybrid Scheme is* better when no contributory protocols are applied (which is expected). However, certain combinations of non-contributory with contributory schemes reduce the performance more than certain combinations of two non-contributory protocols do. Among undeniable winners for most cases are combinations of OFT and TGDH protocols with other efficient protocols with respect to a particular cost. Generally, the following is true:

Different combinations of protocols may prevail for different costs depending on the partial performance of each protocol for the particular cost.

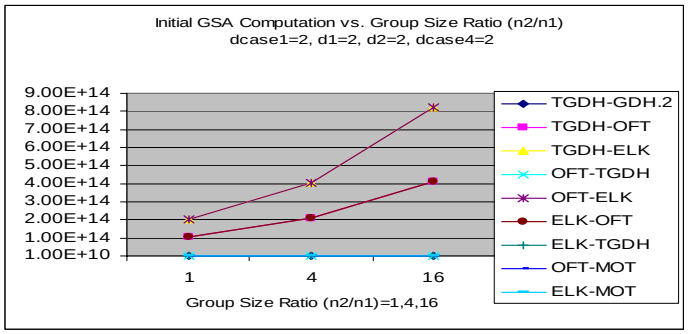

Figure2: Init. GSA Comp/tion cost vs. Group Size Ratio (n2/n1):TGDH-MOT, OFT-MOT, ELK-MOT, TwoTree, SingleTree, followed by GKMP-GKMP, OFT-OFT reduce cost the most. Combinations with ELK at 2nd level worst

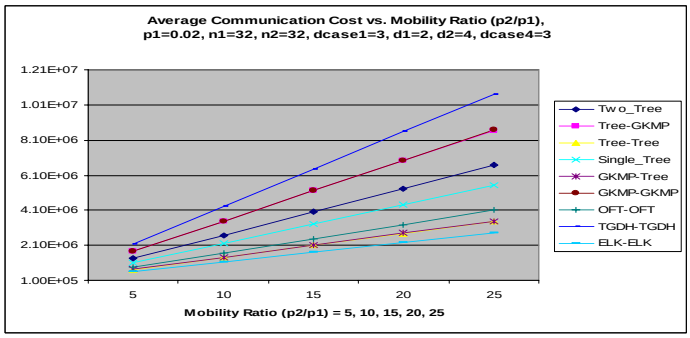

Figure 3: Avg Comm/tion cost vs. Mobility Ratio (p2/p1): combinations with OFT and ELK at 2nd level, followed by GKMP-GKMP, Tree-GKMP reduce avg. Comm/tion the most. Combinations with TGDH, Tree-GKMP, Tree-Tree worst.

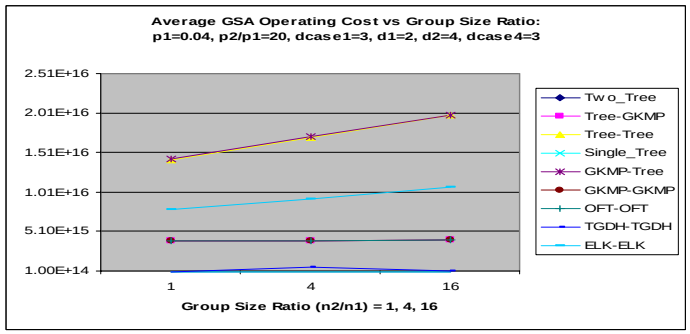

Figure 4: Avg GSA Operating cost vs. Group Size Ratio (n2/n1): SingleTree, TwoTree, followed by combinations of TGDH reduce cost the most. GKMP-Tree,Tree-Tree, and combinations with OFT or ELK at 2nd level are the worst.

Example: The first six schemes combine *CBT* and *GKMP* protocols in different ways [11]: For all costs of the *Two-Level Hybrid Scheme* but for the avg. Comm/tion, *GKMP-GKMP* and *Tree-GKMP* clearly prevail. For the avg. Comm/tion costs however *Tree-Tree* and *GKMP-Tree* are the winners. For GSC Initial and Operative costs *SingleTree* presents the worst behavior (since it is loaded with the tasks of GSAs), but presents much better behavior along with *TwoTree* with respect to Comm/tion costs (it spares comm/tion between GSC and GSAs)

CONCLUSION

The paper discusses the framework and constraints under which existing and novel protocols can become scalable, robust and efficient in MANETs. The diversity in MANET environments and in the nature of key distribution protocols motivates us to classify them in categories and deal with each problem separately. We address key distribution in a *Flat Oriented MANET* (no trusted special hubs assumed) by introducing two novel hybrid protocols MO and MOT-based on the original 2^d-Octopus (O). All three are described in detail in our references and cost functions in terms of communication and computation are derived for all operations. From our performance evaluation, we show that MOT outperforms (O) in all cases, and OFT as of computation costs. As far as Communication Addition/Eviction costs, MOT gets closest to OFT than all the rest. For the *Military Oriented MANETs*, we developed a Hierarchical Hybrid Scheme that exploits the best possible way the diversity of the battlefield. It models most of these environmental variations so that the first level of the scheme represents nodes at the higher end, and the second level nodes at the lower. This scheme allows for great diversity of protocols applied to both levels, given the circumstances. It is robust and very efficient.

REFERENCES

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

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

[3] Maarit Hietalachti. Key Establishment in Ad-Hoc Networks. Master's Thesis. Helsinki University of Technology. Laboratory for Theoretical Computer Science Finland. May 2001.

[4] Adrian Perrig. Efficient Collaborative Key Managements Protocols for Secure Autonomous Group Communication. International Workshop on Cryptographic Techniques and E-Commerce CrypTEC '99.

[5] N. Asokan and Philip Ginzboorg. Key-Agreement in Ad-Hoc Networks. Elsevier Preprint. 23: 1627-1637, 2000.

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

[7] D.A. McGrew, and A. T. Sherman. Key Establishment in Large Dynamic Groups Using One-Way Function Trees. Technical Report No. 0755, TIS Labs at Network Associates, Inc., Glenwood, MD, 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 Engineering Task Force. July 1997.

[10] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. Technical Report 2, USC Technical Report 00-737, August 2000.

[11] Maria Striki, John S. Baras. Key Distribution Protocols for Multicast Group Communication in MANETs. CSHCN Technical Report 2003