New Algorithms for the efficient design of topology-oriented Key Agreement Protocols in Multi-hop Ad Hoc Networks

Maria Striki
Telcordia Technologies Inc.,
One Telcordia Drive
Piscataway, NJ, 08854, USA

John S. Baras,
Institute for Systems Research
University of Maryland, College Park
College Park, MD, 20742, USA

Kyriakos Manousakis
Telcordia Technologies Inc.,
One Telcordia Drive
Piscataway, NJ, 08854, USA

Abstract-**Securing group communications in resource constrained, infrastructure-less environments such as Mobile Ad Hoc Networks (MANETs) has become one of the most challenging research directions in the areas of wireless network security. MANETs are emerging as the desired environment for an increasing number of commercial and military applications, addressing also an increasing number of users. Security on the other hand, is becoming an indispensable requirement of our modern life for all these applications. The inherent limitations of such dynamic and resource-constraint networks impose major difficulties in establishing a suitable secure group communications framework. This is even more so for the operation of Key Agreement (KA), under which all parties contribute equally to the group key. The logical design of efficient KA protocols has been the main focus of the related research to-date. Such a consideration however, gives only a partial account on the feasibility and performance of a KA protocol in a multi-hop network. This is because protocols have been evaluated only in terms of the group key related messaging in isolation from the underlying network functions that interact with the logical scheme (i.e. routing). In this work, we contribute towards efficiently extending a number of Diffie-Hellman (DH)-based group KA protocols in wireless multi-hop ad hoc networks, and measuring their performance over these networks. Towards this end, we introduce a number of new algorithms that merge the logical design of KA protocols with the underlying routing and produce protocols that substantially improve one or more metrics of interest. Indeed, the resulting protocols are significantly more efficient in some or all of the above metrics, as our analytical and simulation results indicate.**

***Index Terms*—Key Agreement, Diffie Hellman, Approximations**

I. INTRODUCTION

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. As the development of multicast services such as secure conferencing, visual broadcasts, military command and control grows, the research on security for wireless multicasting becomes increasingly important. The role of key management (KM) is to ensure that only valid members have access to a valid group key at any time. It is essential to develop a secure, robust KM scheme for group communications in these environments. The operation of Key Agreement (KA) is a subset of the broad KM functionality and imposes that all participants contribute (almost) equally to the group key establishment, by interacting among themselves in ways designated by the specific protocol applied. Compared to other tasks classified under KM, such as this of key distribution, KA operates on an inherently more complicated communication regulation mechanism. The characteristics of MANETs constitute the major challenge for the design of suitable KM schemes and have even more severe impact on the operation of KA. We are dealing with dynamic, infrastructure-less networks of limited bandwidth, unreliable channels where topology is changing fast. Network nodes have limited capacity, computational or transmission power. Connections are temporary (mobility changes, battery drainage, poor physical protection) and unreliable. These constraints turn most of the existing protocols inefficient in MANETs. Along with the continuous quest for the design of more efficient schemes than the existing ones, the need for the new KA schemes to handle successfully and tolerate network dynamics and failures with low impact in a network with large number of nodes is now equally important. Upon failures and disruptions, it is often the case that a KA protocol must restart the group key establishment process from scratch. Whenever this event occurs (may be too often), the underlying routing is invoked once again, a significant amount of relays become involved in the exchange of large keying messages, and considerable delay (reflected in the total number of rounds for the successful termination) burdens the network. The overall performance of the protocols degrades even more because of the indirect impact of excessive routing on additional network layers (i.e. QoS deteriorates due to more collisions at the MAC layer, the bandwidth usage and the consumption of network resources increase undesirably). In MANETs, bandwidth and power consumption are valuable resources, and nodes cannot afford to waste. For all these reasons, reducing the combined costs resulting from the routing and communication exchanges among nodes becomes essential if we want to apply more sophisticated KA schemes on a resource-constrained MANET.

The logical design and analysis of efficient KA protocols has been the main focus of the related research to-date. Such a consideration[1] however, gives only a partial account on the feasibility and performance of a KA protocol in a wireless multi-hop network. This is the case because the evaluation of

[1] Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance (CTA) Program, Cooperative Agreement DAAD19-2-01-0011. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

the protocols is conducted via a logical abstraction of the underlying physical network in such a way that essential inseparable otherwise operations, such as the underlying routing, are left out.

In this work, we efficiently extend a number of Group Diffie-Hellman (GDH) Key Agreement (KA) protocols on wireless multi-hop ad hoc networks (i.e. GDH.1, GDH.2, ING), and measure their performance over these networks. More specifically, we estimate the combined routing-communication costs of the KA protocols under study, taking into account the underlying routing. Then, we optimize or improve these protocols by means of establishing specific communication schedules that are subject to the underlying routing every time. We show that the modified protocols substantially improve one or more metrics of interest. We distinguish two cases:

1) the graph that consists of group members represents a physical topology, i.e. each edge is a physical link, and

2) the graph that consists of group members represents a logical topology, i.e. each edge is a logical link, bounded however from the number of hops between two vertices, as provided by the underlying routing. In this case, intermediate hops are not necessarily group members.

The original protocol versions do not exploit members' topological proximity in general. A pre-agreed schedule is used, based on members' attributes, like their "hashed" *IP*s or *ID*s, but not on the underlying routing. After our investigation of the protocols subject to the underlying routing, we can see that the routing structure of each protocol poses a different optimization problem (usually *NP*-complete) for each of the given metrics. Given that, we focus on efficient approximations towards generating "routing aware" communication schedules that significantly improve the performance of the discussed schemes. In our previous work [23], we imposed that the secure group members graph is connected (i.e. the logical graph coincides with the physical graph), as described in case 1. In this paper, we are addressing far more generic and realistic scenarios (networks where the group member graph is not rigorously connected, i.e. non-member relays may be included in the path of two connected members), as described in case 2. Our newly introduced heuristics for the establishment of modified communication schedules that are subject to the underlying routing achieve much better approximations of the metric functions we want to optimize. The comparison between the new routing-topology oriented and the original schemes is done via simulations.

In section 2 we give an overview of related work on KA and in section 3 we describe the original schemes. In section 4 we provide an outline of both our previous and current work. We motivate our transition from case 1) to case 2) and point out the differences in the associated models and specifications. In section 5 we provide a detailed overview of our new algorithms and heuristics. In section 6 we present the performance analysis of our auxiliary framework. In section 7 we present the simulations set-up and our simulation results which clearly point out the superiority of these new heuristics, and attest to the validity of our analytical results regarding the auxiliary framework. Finally, in section 8 we conclude the paper.

II. Related Work

Proposals related to secure group KA protocols abound in the literature, and can be usually found under the broad category of contributory schemes. Most of them correspond to a logical consideration in terms of design, or address wire-line networks and cannot operate as such in MANETs.

Becker et al. [1], derived lower bounds for contributory key generation systems for the gossip problem and proved them realistic for Diffie-Hellman (DH) based protocols. They used the basic DH distribution [3] extended to groups from the work of Steiner et al. [2], where three new protocols are presented: GDH.1-2-3. Ingemarsson et al. [4] presented another efficient DH-based KA scheme, "ING", logically implemented on a ring topology. In addition, Burmester et al. [5] introduced a new GDH protocol, denoted as BD (very efficient in terms of round complexity). Another hybrid DH-based KA scheme is TGDH introduced by Kim et al. [7]. It is an efficient protocol that blends binary key trees with DH key exchanges. Becker in [1], introduced Hypercube, that requires the minimum number of rounds. Becker also introduced Octopus scheme as one that requires minimum number messages and then derived the 2^d-Octopus that combined Octopus with Hypercube to a very efficient scheme that works for arbitrary number of nodes. Related work can be found in [6, 8, 9] as well.

There exist some more recent proposals of KA for wireless ad-hoc networks. Even these, do not seem to scale well or handle successfully the network dynamics. Some of these approaches rely on public key cryptography, which is very expensive for resource constrained nodes, or on threshold cryptography [14-19], which results in high communication overhead, and does not scale well. Katz et al. [10, 11], improve on existing KA schemes either by rendering them more scalable or by enhancing their security against various kinds of attacks. Still, the described algorithms are implemented on logical graphs, or address wire-line networks. Amir et al. [12, 13], focus on robust KA, and attempt to make GDH protocols fault-tolerant to asynchronous network events. However, their scheme is designed for the Internet, and requires an underlying reliable group communication service and message ordering, so that preservation of virtual semantics is guaranteed. In [20], it is shown that the consideration of the physical location of members is critical for developing energy-efficient KM schemes, and based on this observation a new energy-aware KM scheme is proposed. In [22], additional Octopus protocols for robust and efficient group communications in MANETs are proposed. The primary focus of this work is the logical evaluation of the proposed schemes, in isolation from network functions that interact with them. In [23], we study the extension of a number of known KA protocols over multi-hop ad hoc networks. The investigation conducted sets the foundations for our current study, but it is quite preliminary as it addresses a rather specialized network environment, and there seems to be much scope for improvement anyway.

III. Original GDH-Based Schemes (Overview)

Even though the original versions of the logical design of the GDH schemes we are studying are well documented in our

references [2, 3, 4, 5], we give a more detailed and simplified description of their basic operation in this section, in order to make it easier for the reader to follow the next sections.

Notation_1*:* Let $B(x) = a^x$ be the blinding (exponentiation under base a) of value x and let $\varphi(x) = x \bmod n$. We assume that all exponentiations are modular exponentiations (MEs). In this analysis, the modular reduction $\varphi(x)$ of a secret value x, prior to its blinding, $B(\varphi(x))$, is implicitly assumed, but not reflected to our equations, for ease of the notation. Hence, we replace the expression $\varphi(B(x))$ with $B(x)$.

GDH1: This protocol assumes that all parties are connected according to a logical Hamiltonian path and consists of two stages: up-flow (collecting members' contributions) and down-flow (allowing all members to compute the common group key K_n). Any member M_i is associated with an individual secret key N_i. Each member M_i receives a collection of (i-1) intermediate values from member M_{i-1}. The task of each member M_i on the up-flow is to compute $a^{N_1 \ldots N_i}$ by raising $a^{N_1 \ldots N_{i-1}}$ - the highest numbered incoming value – to the power of N_i, append it to the incoming flow and forward all to M_{i+1}. For example, member M_4 receives from member M_3 the following set: $\{a^{N_1}, a^{N_1N_2}, a^{N_1N_2N_3}\}$ and forwards to member M_5 the modified set: $\{a^{N_1}, a^{N_1N_2}, a^{N_1N_2N_3}, a^{N_1N_2N_3N_4}\}$. In the up-flow, each member performs one ME and the message between members M_i and M_{i+1} contains i intermediate values. The final transaction in the up-flow stage takes place when the highest-numbered member M_n receives the up-flow message and computes $(a^{N_1 \ldots N_{n-1}})^{N_n}$ which is the intended group key K_n. After obtaining K_n, member M_n initiates the down-flow stage. In this stage, each member M_i executes i MEs: one to compute K_n and (i-1) to provide intermediate values to subsequent (lower-indexed) group members so that they can compute the group key themselves. For example, assuming n=5, member M_4 receives from member M_5 the following down-flow message: $\{a^{N_5}, a^{N_1N_5}, a^{N_1N_2N_5}, a^{N_1N_2N_3N_5}\}$. First, member M_4 uses the last intermediate value to compute the group key K_n. Then, it raises all intermediate values to the power of N_4 and forwards the resulting set: $\{a^{N_5N_4}, a^{N_1N_5N_4}, a^{N_1N_2N_5}, a^{N_1N_2N_5N_4}\}$ to M_3. In general, the size of a down-flow message decreases on each link; a message between M_{i+1} and M_i includes i intermediate values. Each member M_i performs i MEs: one to compute K_n and (i-1) to provide intermediate values to lower indexed members, by raising them to the power of its own secret exponent. The size of the down-flow message decreases on each link, as a message between M_{i+1} and M_i includes i intermediate values.

GDH2: In order to reduce the total number of rounds, GDH.1 is slightly varied, so that: *a*) in the up-flow stage each member M_i needs to compose i intermediate values (each with i-1 exponents) and one cardinal value with i exponents; member M_n is the first member to compute the group key K_n and the last batch of intermediate values, *b*) in the down-flow stage, the last party on the path or highest indexed member M_n broadcasts the (n-1) intermediate values to all group members. It is assumed that all parties are connected through a logical Hamiltonian path, and member M_n can reach all others using a broadcast channel. Each member M_i raises a corresponding intermediate value to its own secret value N_i and reconstructs the group key. For instance, considering a group consisting of four members, member M_2 uses the following intermediate value among the rest received from member M_4: $a^{N_1N_3N_4}$, and computes the final group key: $K_n = (a^{N_1N_3N_4})^{N_2} = a^{N_1N_2N_3N_4}$.

ING: The protocol is executed on a virtual Hamiltonian ring. It requires that all parties are connected according to a logical ring, and completes in (n-1) rounds after a synchronous start-up. In any round, every party raises the previously-received intermediate key value to the power of its own random exponent and forwards the result to the next party. Hence, in every round, n message exchanges are being executed. After (n-1) rounds and n(n-1) message exchanges every member computes the same group key $K_n = a^{N_1N_2 \ldots N_n}$. The following formula shows the algorithmic flow of ING for round k, where $k \in [1, n\text{-}1]$: $M_i \rightarrow a^{\Pi\{N_j | j \in [(i-k) \bmod n, i]\}} \rightarrow M_{i+1}$.

IV. Previous and Current Approach: Network Model, Specifications and Requirements

In this section we provide an overview of our previous work on this topic. We describe the network model, the requirements and assumptions for the methods used in [23], and we justify our motivation for extending this work to accommodate our new objectives and directions. Our previous work will serve as a starting point for our current approach. We focus on the differences in the two methodologies and discuss our need to look for more general and efficient approximations than those previously introduced.

Notation_2: The prefix "*nt*" abbreviates the extension of the discussed KA schemes on a wireless multi-hop network with "no topology" or else underlying routing considerations, and the prefix "*wt*" (i.e. with topology or else underlying routing considerations) abbreviates the topologically oriented extensions respectively.

Notation_3: Let n be the number of members in the secure group, and m be the number of nodes in the network (size S). Also, let D be the diameter $diam(G)$ of the network graph G; that is, the max number of hops between a pair of nodes in V. Also, let $R(N_i, N_{i+1}) = R_{i,i+1}$ be the number of hops in the path between two members N_i and N_{i+1}, where $R_{i,i+1} \leq D$.

Notation_4: K is the bit size of an element in the algebraic group used (where the decision DH problem is assumed hard).

A. Existing Approach Setting and Results (Starting Point)

In [23], we re-evaluated the KA protocols discussed by using a natural implementation of routing and broadcast/unicast operations; specifically, by executing the protocols blindly on a

network where multi-path routing is required for group members to communicate, and where not all members can be directly reached via a single broadcast. We ran those protocols on top of this framework using a communication schedule that is based merely on arbitrary member *ID*s. This "*nt*" approach may lead to excessive routing and produce unnecessary relay nodes, and consequently high communication cost, as we can also see from Table I of our relevant results. These results are very indicative of the actual communication overhead and rounds that will be incurred from these protocols when applied on a multi-hop network, even if the associated logical key generation algorithm appears to be very efficient (the original versions of the studied KA protocols are merely the key generation algorithms run on a logical framework).

TABLE I
PERFORMANCE OF KA PROTOCOLS WITHOUT TOPOLOGICAL ASSUMPTIONS

	Logical *Lt*	Logical *CCost*	*nt-Lt*	*nt-CCost*
ING	n-1	$n(n$-1)	*D(n-1)*	*Dn(n-1)*
GDH1	2(n-1)	$n(n$-1)	*2D(n-1)*	*Dn(n-1)*
GDH2	n	(n-1)(n/2+2)	*Dn*	*Dn(n-1)/2*

Table 1: Performance of: (*a*) KA protocols over logical networks, (*b*) *nt*-extension of KA protocols over multi-hop ad hoc networks.

We then tried to improve the efficiency of each protocol by exploring the potential of optimizing their combined routing and communication costs with the use of a *wt* simulation of the logical network over an arbitrary network graph *G*. In other words, we integrated the protocols with new communication schedules subject to the underlying routing. As a first step, we used a rather restraining model of the connectivity among group members: the secure group graph is defined as $G(V,E,w)$, where $E \in V \times V$, and w is the edge weight function that maps any edge e in G to an integer, i.e. w: $(i,j) \rightarrow Z^*$, $e(i,j) \in V \times V$. An edge between any two members (i, j) exists if and only if the members are within each other's radio range (bidirectional connectivity). We impose that these members have distance of 1-hop, or equivalently $w(e_{ij}) = 1$. Hence, any direct link has weight 1, and any path that connects two members indirectly consists only of group members. For ease of the notation we set: $G(V,E,w) = G(V,E,1) = G(V, E)$.

We further assumed that each message from a member in *G* is timely sent (i.e., there is no congestion) and reliably and timely received by all neighbors. Our schemes inherit the same security properties as those of their original ancestors. Our main objective was to meet efficiency requirements of low communication overhead and latency for the group key establishment during the initial state of key generation. We assumed that the underlying routing is capable of establishing end to end paths, avoiding intermediate link failures. We did not consider dynamic cases (i.e. link failure, mobility) under which the network could be partitioned. The behavior of the protocols at steady state (i.e. re-keying operation) was not considered within the scope of that work.

Under the *nt* communication schedule, members' placement and consequently the routes formed are random. Under the worst case scenario the latter routes could be forming a bipartite graph with links of *D* hops each, which is far from the optimal. This arbitrary factor that emerges when we merge the key generation algorithm "blindly" with the underlying routing is what we try to capture, model, and quantify with our analysis. In summary, we defined and later focused on minimizing the following 6 quantities, or performance metrics (scaled down by *K*):

Communication Cost (*CCost*):

$$CCost_1 = (n\text{-}1) \times \sum_{i=1}^{n} R_{i,i+1} \quad (1).$$

$$CCost_2 = \sum_{i=1}^{n-1} 2 \times i \times R_{i,i+1} \quad (2).$$

$$CCost_3 = (n\text{-}1) B_n \quad (3).$$

Latency (*Lt*):

$$Lt_1 = \sum_{i=1}^{n} R_{i,i+1} \text{ , or } (n\text{-}1) \times max\{ R_{i,i+1} \} \quad (4).$$

$$Lt_2 = 2 \times \sum_{i=1}^{n} R_{i,i+1} \quad (5).$$

$$Lt_3 = \text{max\_length } (B_n) \quad (6).$$

Each protocol poses two different optimization problems as its routing structure defines a specific optimization function for each of the two metrics of **latency (*Lt*)** and **combined communication cost (*CCost*)**. Given these quantities, we developed bounds for the two metrics of each protocol as follows: ING: from (1) and (4), GDH.1: from (2) and (5), GDH.2: from (2), (3), (5), and (6).

The solutions to most of these functions can be mapped to *NP*-complete problems, e.g. (1), (4), (5) are mapped to the Traveling Salesman Problem (TSP). Thus, finding approximations of the optimal solutions to these quantities, result in more efficient metrics for the protocols.

We then proposed a number of heuristics (details can be found in [23]), most of which rely on an auxiliary framework that includes the generation of a tree that spans all *n* members of the secure group. This spanning tree (ST) has the following property by definition of the associated network graph *G*: the weight of any link that directly connects any two tree members is 1. In this case, this ST is in fact a minimum ST (MST). This equivalence allows us to use some approximations based on the existence of an MST over the group members. We briefly quote all the approximation used below:

1) *Solution to (1), (4), (5): Full Walk on the MST*
2) *Solution to (2), (4): Extended ING ring with dilation 2*
3) *Solution to (2): Closest Point Heuristic*
4) *Solution to (3), (6): Broadcast Tree*

Indeed, by using these approximations to set up a *wt* schedule, the performance of the studied KA schemes was significantly improved by at least a factor of *D* (or *n*) in most cases, in contrast to the arbitrary execution of the *nt* protocol versions.

B. Current Approach Setting, Requirements and Objectives

The results illustrated in [23] are just the beginning of this challenging research problem. There is still ample scope for improvement, as long as all these solutions are just approximations. More importantly, the model used in [23] is

very particular and does not realistically represent the general multi-hop ad hoc network, where the path between two members may include non-member relays too. Our objective in our current work is to continue our investigation on more efficient approximations, on a far more general network model however, the specifications of which we provide next.

We maintain the assumptions of the previous section except for the model of the network graph, with which we essentially delineate our methods and solutions from the previous. In our network model we allow non-member relays as well in the routing path between two group members. We assume a generic Dijkstra routing protocol with the property that it always finds the shortest paths between members. Through the underlying routing, each member obtains the routing path(s) to its closest neighbor(s). We dynamically determine the upper limit in the number of hops between members considered immediate "logical neighbors" (proximity). If no neighbors are found in the proximity, the search diameter (TTL) is gradually expanded until a pre-agreed number of such neighboring members are found or until the TTL threshold is reached. So, two members i, j are considered "virtually connected" if they share at least a routing path $R_{i,j}$ with length up to a pre-defined threshold that depends on the TTL, i.e. $R_{i,j} \leq Th\ (TTL)$. Assuming that a direct link between two network nodes has weight 1, the virtual link between two virtually connected members may take any value $x \leq Th\ (TTL)$. Hence, the secure group connectivity graph is defined as $G(V, E, w)$, where $E \in V \times V$, and w is the edge weight function that maps any virtual edge in this graph to an integer, so that w: $(i, j) \rightarrow Z^* \cap [1, Th(TTL)]$, $e(i, j) \in V \times V$. The weight of each link is the distance between the two group members in terms of number of hops (relays). We are interested in minimizing the combined routing and communication cost (*CCost*), resulting from the execution of the KA protocols discussed. Depending on the metric we want to optimize, we can adjust the link weight accordingly: it may include delays, traffic, robustness, mobility, geometric distances, etc.

We now examine if it makes sense to apply the methods proposed in [23] to our new setting. In [23] we generated a ST formed strictly by the n group members. The value of any link weight was always 1, and an MST would coincide with any ST. Implementing for all three protocols a full walk over an MST guarantees solutions that are better or equal than twice the optimal. The optimal in that case was the size of the secure group n, which was indeed what we were after. Since the size of the secure group is assumed to be much smaller than the size of the network, the above solutions are still acceptable for the metric functions of the protocols. In fact, we witnessed a great improvement in the performance of the protocols with these approximations. However, in the current setting, the same approximations would provide us with solutions that are better or equal than twice the size of the virtual ST, or MST with arbitrary weights, which could be the size of the whole network in the worst case scenario. The same is the case with the technique that uses dilation of group members, which becomes preventive for a growing member or network graph. Hence, the larger the network the more inefficient the aforementioned method becomes. Moreover, the edge weights between two virtually connected tree members depend on the size of the minimum path shared which is most likely greater than 1, unless there is a direct link between two members. In this case, *a ST is no longer equivalent to an MST*. So, *in our new setting we will need to compute an MST* and not just any ST. In fact, *we will need to find the MST of all MSTs originating from every single group member*, if we want to apply an equivalent to our previous method on the discussed KA protocols. Obviously, utilizing a core framework that computes the MSTs from all group members is an over-kill. This is most discouraging factor that prevents us from applying blindly the previous techniques to our new setting. At the same time, we seek lightweight solutions, so that the extra overhead required for the generation of the backbone framework is maintained as low as possible. Again, we stress the importance of limiting the total number of relays required for the execution of the KA protocols. The less the total number of relays, the less the side effects of routing on the standard communication are.

V. Overview of New *WT*-GDH KA Protocols

From the previous discussion, it becomes obvious that there is a need for far more efficient communication scheduling methods than merely this of the MST approximation. In what follows, instead of computing the cost associated with extracting the MST of all MSTs originating from any group member, we describe and evaluate a very efficient novel heuristic for extracting a "low cost" Hamiltonian path from only one MST originating from a single group member. This algorithm applies to all the discussed KA protocols that require a ST to derive a *wt* efficient communication schedule. We show that with the appropriate manipulation of the generated MST, we achieve even further improvement on the metrics of interest: *CCost*, *RCost*, and *Lt*. The algorithm unfolds in two steps: MST generation and MST manipulation.

A. MST Generation

We generate our MST starting from member A without loss of generality, by applying a distributed version of *Prim's* algorithm. The root member can be one that acquires certain attributes (i.e. pre-selected group leader, pre-agreed initiator, member with the highest residual energy, etc.). *Prim's* method [25] is based on a greedy strategy, captured by a "generic" algorithm which grows the MST one edge at a time. The algorithm manages a set of edges H, maintaining the following loop invariant: prior to every iteration H is a subset of some MST. *Prim's* algorithm has the property that the edges in H always form a single tree. This strategy is greedy since the tree is augmented at each step with an edge that contributes the minimum amount possible to the tree's weight. In order to implement this algorithm as such, all members must have global information of the link weights of all other members, so that they all see which member in H has the minimum link weight and allow the growing MST to expand towards this direction. We adjust this algorithm to our distributed environment, by having each member that joins H report its candidate links to the root A. At each step, the root determines the next member J to join H by examining all unused candidate links of all members that currently belong to H. Then, the root

sends a *Join Flag* to member *J*, *J* joins *H*, and so on and so forth. It has been shown that the improved running time of the original *Prim's* algorithm is:

$$Lt1 = \mathrm{O}(E + V\log_2 V), \quad (7).$$

Notation_6: Let the weighted path between any member *J* and the root *A* be denoted as $R_{J,A}$ (or $R_{J,Rt}$). Also, let PK_S denote the bit size of the packet that carries a member's candidates' information up to the root, and let K_S denote the bit size of the *Join Flag* from the root to the next member that joins *H*. Since K_S is the size of a control packet, we safely assume that the following inequality holds: $K_S << K$.

The combined communication cost incurred to the network from the generation of the distributed MST becomes:

$$CCost1(aux) = \sum_{j\in V} R_{j,A} \times (PK_S + K_S)$$
$$CCost1(aux) \approx |V| \times (PK_S + K_S) \times \mathrm{avg}(R_{j,A})$$
$$CCost1(aux) \approx \frac{1}{2}|V| \times (PK_S + K_S) \times \max_j(R_{j,Rt}) \quad (8).$$

The associated latency or running time for the adjustment of *Prim's* algorithm becomes:

$$Lt2 = 2 \times \sum_{j\in V} R_{j,A} \approx 2 \times |V| \times \mathrm{avg}(R_{j,Rt})$$

$$Lt2 \approx 2 \times |V| \times \frac{1}{2} \times max_j\,(R_{j,Rt}) \approx |V| \times max_j\,(R_{j,Rt}). \quad (9).$$

The running time of the adjusted algorithm in total becomes:

$$Lt1 + Lt2 = \mathrm{O}(E + V\log_2 V) + |V| \times max_j\,(R_{j,Rt}),$$
$$Lt1 + Lt2 = \mathrm{O}(E + V\,(\log_2 V + max_j\,(R_{j,Rt}))) \quad (10).$$

B. MST Manipulation

Until now, we have generated an MST starting from member *A*. A full walk on this MST will provide us with a Hamiltonian path or tour with cost $C_{MST} < 2C_{OPT}$. We will investigate if we can do better than that for GDH.1, GDH.2, and ING that are logically deployed on a Hamiltonian path or ring respectively. Thus, we look for heuristics that produce better results than the *full walk on an MST*. Although the latter is described in [23] in detail, we briefly summarize it here before we proceed with the heuristic description, for facilitating the reader in what follows.

Pre-order traversal of an MST: Using an MST to approximate the *Traveling Salesman Problem* (TSP) with *triangle inequality* provides a near-optimal tour of a complete undirected graph, designating a routing path at most two times the optimal. The *triangle inequality* implies that going by way of any intermediate stop cannot be less expensive. A full walk of the MST traverses every edge exactly twice. The expression in (2) is shown to be minimized if the following inequality holds: $R_{n-1,n} \le R_{n-2,n-1} \le .... \le R_{1,2}$. In GDH.1-2, the number of messages communicated successively in the up-flow (parameter *i*) is incremental. Hence, the routing paths of successive members should be selected to be non-increasing. If we fix the GDH.1 *backward schedule first,* so that the first edge selected in the MST is assigned to relay the maximum number of messages (*n*-1), the second edge to relay (*n*-2) messages, etc., the "non-incremental" requirement is satisfied. We want to identify the *appropriate* traversal method to visit all vertices of a ST and establish the GDH.1 *backward schedule first*. By examining the common traversal methods (*pre-order*, *in-order*, *post-order*), we select the *pre-order* tree walk. An intuitive reason for this is that a *pre-order* tree walk prints (visits) the root before the values in either sub-tree. So, it uses a rather greedy approach by adding the "best nodes" first, in a forward manner, the earliest possible in the backward schedule. On the contrary, the nature of the *in-order* and *post-order* walks is such that recursion prevents the immediate selection of the shortest paths in the tree between successive in the schedule nodes. By visiting the vertices indicated by the *pre-order* walk backwards, we obtain the forward GDH.1 schedule.

Heuristic Description: We start with the following observation: *if the generated MST was in fact a chain, then the desired Hamiltonian path would be directly provided and would result in the same cost as this of the MST*. Hence, the more the resulting MST resembles a single chain, the less the cost of the resulting Hamiltonian path (not tour) is expected to be. Based on this observation, we initiate the manipulation of the MST with the following *transformation*: During the formation of the MST the *two longest distinct paths* from all group members to the root are located. The group member that marks the end of the longest path becomes now the new root of the transformed MST, and the associations between parents and offspring in the existing MST are sequentially altered to accommodate the transformed tree. This process results in *unfolding the MST* to its longest path or else in "*extracting*" the largest possible "path" from the MST (Fig. 1(*b*), 1(*c*)).

Next, each member that belongs to the new ST, recursively rearranges its offspring in the order of decreasing distances (number of hops including relays) from their tree leaves (Fig. 1(*d*)). It is obvious that the backbone of the tree, which is the previously unfolded path, will be accessed first by a pre-order tree traversal. In particular, in Fig.1(*d*) the *pre-order* tree traversal is defined so that it visits the root before the values in either sub-tree and then proceeds from the *leftmost* offspring towards the *rightmost* one. It can be directly seen that in the case we want to generate a Hamiltonian path from this ST tree, *all members that belong to the "unfolded" path will be visited only once*. No recursions occur on the unfolded path. Hence, the longer the unfolded path is, the less the number of members that will be revisited is. Consequently, this modification results in the reduction in the routing overhead for the Hamiltonian path formed. This is also the *intuitive idea* behind the use of this heuristic for protocols GDH.1, GDH.2, and ING.

In the case of GDH.1 and GDH.2, the benefit from having each member in the transformed MST recursively rearrange its offspring is even greater. For the upward stage of GDH.1-2, the MST is traversed as indicated by the previous description of the *pre-order traversal of an MST fixing the backward schedule first*. The MST is also unfolded to its largest path, with the offspring of every member arranged in increasing order of distances. Along the lines of a greedy strategy, we select to fix the backward GDH.1-2 schedule first. For that, we assign each of the *n* group members with a unique sequential *id* from $Z^* \cap [0, n\text{-}1]$. That is, we perform a pre-order visit of the MST by assigning session *id*s from the highest to the lowest one, as we traverse the virtual tree. For example, the first

member encountered in our pre-order walk, say *F*, is assigned with *id* = (*n*-1), the second member encountered, say *L*, is assigned with id = (*n*-2), etc. Then, when GDH.1-2 is executed, member *F* will be the last visited in the upward stage (or the first to initiate the backward), and member *L* will be the predecessor of *F*. By fixing the backward schedule first via a pre-order MST traversal, we ensure that the members carrying the longest KM material are accommodated first. Hence, we still act along the lines of a greedy strategy. *Among siblings,* the following *invariant* is true: the higher the newly assigned *id* of a sibling, the fewer amount of hops (relays) a message originating from this sibling will go through until the destination is reached (successor or predecessor). The new *id* assigned to a member corresponds also to the number of elements (KM data) the member must communicate to its successor or predecessor. Thus, among siblings, the longer the message, the less number of relays it involves. We stress again that we aim in improving the metrics of interest for the studied protocols by manipulating the MST with simple, lightweight, but effective heuristics, like the ones proposed. Below, we briefly summarize our algorithm.

1) Construct MST using distributed Prim (nodes report candidate links to the root which determines the next link).

2) Transform MST by deploying (unfolding) its largest path, modify all parent-offspring associations accordingly.

3) Recursively re-arrange all offspring visited in decreasing distances from MST leaves (case of GDH.1-2).

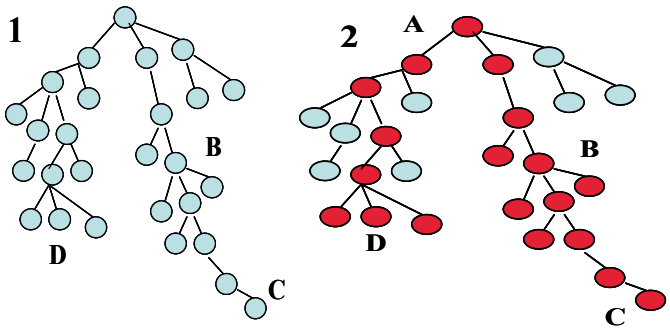


Figure 1. Manipulation of a MST: Transformation by unfolding it to the longest path and recursive re-ordering of each member's offspring.
Figure 1(*a*). Initial MST Prim.
Figure 1(*b*). Identification of the largest path to unfold

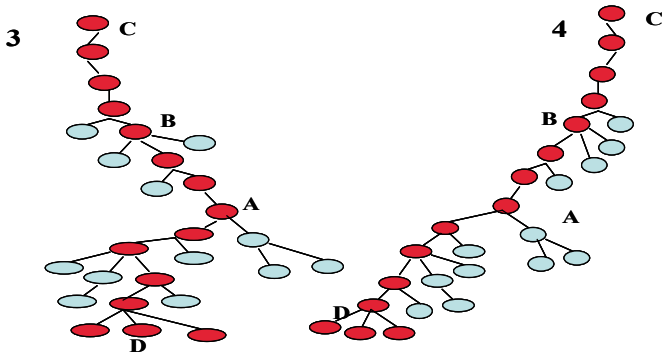


Figure 1(c). MST unfolded to its largest path
Figure 1(*d*). Re-arrange offspring from largest to smallest path.

VI. Analysis of New Auxiliary Framework

The two longest distinct paths are determined along with the initial MST formation. The root stores information about the two members whose distances from the root are currently the longest, and follow two distinct paths to the root. Based on the next candidate that joins the evolving subset *H*, the root updates its related information accordingly. Thus, no extra communication overhead or latency is incurred to the network for this operation. After the two longest paths are determined, the root notifies the member that will serve as the new root, say member *C*. Starting now from *C* all members sequentially alter their parent-offspring association accordingly. This running time for this operation depends on the length of the new unfolded path (UP), i.e. $R_{UP} = R_{C,D}$, as illustrated in Fig 1(*c*).
Then, the associated latency becomes: $Lt3 = R_{UP}$ (11).
The associated communication overhead becomes:
$CCost2\ (aux) = Weight\ (MST) \times K_S = W(MST) \times K_S$ (12).

For the re-ordering process, each offspring recursively notifies its parent of its distance from the related tree leaf. For example, if member *F* collects the maximum distances of all offspring from the tree leaves, it picks the maximum among these values, adds its own link weight towards its parent, and sends this information to its own parent, say member *B*. *B* will collect similar information from all offspring, and so on and so forth. Each parent stores this information and applies a sorting algorithm to its offspring (i.e. QuickSort) to virtually reorder them for the coming pre-order traversal. The sorting calculations can be executed independently from the maximum distance propagation up to the root. The overhead incurred from this operation is the same as before:

$CCost3\ (aux) = W(MST) \times K_S$ (13), and $Lt4 = R_{UP}$ (14).

Notation_7: Let $R_{max} = max_{i,j}(R_{j,\ i})$ be the longest virtual link between any two virtually connected members.

Overall, the communication cost and latency for the generation of the core framework becomes approximately:

$$CCost\ (aux) = CCost1(aux) + CCost2(aux) + CCost3(aux)$$

$$CCost(aux) = \frac{|V|}{2} \times max_j\ (R_{j,\ Rt}) \times (PK_S + K_S) + 2 \times W(MST) \times K_S,$$

$$CCost\ (aux) < \tfrac{1}{2}\ |V|^2 \times R_{max} \times (PK_S + K_S) + 2|V| \times R_{max} \times K_S,$$

$$CCost\ (aux) < |V| \times R_{max} \times (\tfrac{1}{2}\ |V| \times (PK_S + K_S) + 2\ K_S) \quad (15).$$

$$Lt = Lt1 + Lt2 + Lt3 + Lt4$$
$$Lt = O(E + V\ (\log_2 V + max_j\ (R_{j,\ Rt})\ )) + 2 \times R_{UP}\ .$$
$$Lt < O(E + V\ (\log_2 V + |V| \times R_{max}\ )) + 2 \times 2|V| \times R_{max}$$

$$Lt < O(E + |V \log_2 V| + (\ |V|^2 + 4|V|) \times R_{max}\ )) \quad (16).$$

We will now check if the analytical results will be justified through the simulation analysis presented in the following section. We have conducted extensive simulations to measure the auxiliary overhead, the exact communication overhead and latency of the "*wt*" execution of the studied KA protocols. Moreover, we compare these new *wt* protocol versions with the previous *nt*-versions, again by simulations. The results gathered clearly demonstrate the superiority of the *wt* protocol execution.

VII. *WT*-GDH *VS.* ORIGINAL *NT*-GDH SIMULATION RESULTS

A. Simulations Set-Up

We have conducted simulations in order to compare the routing cost of *wt*-(GDH.1, GDH.2, and ING) *vs.* their original versions over ad hoc multi-hop networks. We use various topology graphs to generate the secure subgroups and analyze the performance of the various algorithms. Our network graph represents a single cluster area where a single secure group is deployed. A number of nodes from this graph are randomly selected as group members. The group leader is randomly selected as well. At the end of the group "registration" period, the sponsor piggybacks the list of the legitimate members into the routing packets. We assume a generic Dijkstra routing protocol that finds the shortest paths between members. Through the underlying routing, each member obtains the routing path(s) to its closest neighbor(s). We dynamically determine the proximity with respect to the number of hops between two members. If no neighbors are found in the proximity, the search diameter (TTL) is gradually expanded until a pre-agreed number of members are found.

We further assume that while the backbone framework is being formed, the relative placement of members and consequently the proximity lists do not change significantly. Such a change could result in a different "optimal" solution, and the one currently generated would become outdated and probably suboptimal. However, our algorithm is fairly fast. So, it is not too optimistic to assume that the topological changes that occur do not "offset" our solution much from the target one. It is expected that the higher the mobility of nodes, the worse the performance of our algorithm is. Even though the *wt*-versions are more sensitive to mobility and may be operating under a suboptimal schedule, they still outperform the original schemes in terms of *CCost* and *L*t. On the other hand, the backbone framework can be periodically reconfigured to capture all dynamic changes. The reconfiguration frequency depends on the network dynamics (i.e. mobility, disruptions, battery drainage, etc.), the available network resources, and our own performance requirements. For example, the auxiliary framework may be recalculated whenever the performance of a given protocol degrades to the median of the best execution (the first one after a reconfiguration) and of the average execution of the original *nt* scheme.

For our evaluation, we generated various random graphs for a given input of the number of nodes n and the number of members m. For the same graph and the same input, we have varied the subgroup configuration, i.e., we have selected the n members in a random manner. For each graph of input $<n, m>$ and for each subgroup configuration, we have evaluated the three metrics of interest (*CCost*, *RCost*, *Lt*) of the *wt*-versions *vs.* the *nt*-versions (original), and we have averaged the results for all random graphs with the same inputs $<n, m>$. We have tested the following cluster-subgroup scenarios:

Cluster Size: [100,…, 600], **Subgroup Size**: [8,…64].

B. Simulation Results:

We illustrate in the following graphs some indicative results on the communication and routing overhead produced by the studied protocols and their new *wt* versions, measured in terms of the total number of hops required for the protocols to successfully terminate. The KM messaging is very heavy for the network nodes, so our aim is to reduce the overall number of bits (or packets) required and relieve as many nodes as possible from "relaying" large keying data.

The following graphs reflect a number of important metrics that justify the value of our new *wt*-algorithms: (*a*) the combined communication overhead (*CCost*) and routing overhead (*RCost*) of the *wt*-versions *vs*. the existing ones, and (*b*) the communication and computation overhead for the auxiliary framework (*CCost*(*aux*), *CompCost*(*aux)*), under various scenarios of secure group and network size. The graphs in case (*a*) are all scaled by a factor K, while those in (*b*) represent actual values of the auxiliary framework.

The routing and communication overhead is significantly reduced in all the scenarios captured by our simulations. Indeed, the new schemes results in significant savings in terms of routing (and consequently communication) overhead, and in many cases the associated ratio is:

$$R_{COMM} = \frac{CCost(ING\_Opt, n, S)}{CCost(ING, n, S)} \approx \frac{1}{2}.$$

We illustrate the above with an indicative arithmetic example: for a subgroup of size 32, and a network of size 200, the averaged relays produced (*RCost* or equivalently *CCost*) are 1215 for ING_Opt (i.e. with *wt* schedule), and 2595 for ING (i.e. original with *nt* schedule), and hence $R_{COMM} < \frac{1}{2}$.

Furthermore, we are able to verify that the communication and computation costs resulting from the auxiliary framework of the *wt*-versions match our analytical results. Indeed, let us for example recall the analytical formula we derived for estimating the *CCost* (*aux*) metric:

$$CCost\ (aux) < |V| \times R_{max} \times (\tfrac{1}{2}|V| \times (PK_S + K_S) + 2\ K_S) \qquad (15).$$

Assuming that the maximum number of neighbors for each member is 10-12 (verified from our simulations as well), and having set $K_S = 8$, we select $PK_S = 8 \times 12 = 96$ (bit size of control packet × maximum number of neighbors) Also, we select $R_{max} = 8$, and obtain the following expression for (15):

$$CCost\ (aux) < 8 \times |V| \times (52 \times |V| + 16).$$

Having evaluated this expression for a subgroup of 32 members we obtain: *CCost* (*aux*, V=32) < 430,080 bits. Indeed our results verify that this upper bound holds, since some indicative values of *CCost* that we obtain for group size n and network size S are the following: *CCost* (*aux*, n=32, S=200) = 336,977 bits, *CCost* (*aux*, n=32, S=300) = 395,881 bits, *CCost* (*aux, n*=32, S=400) = 256,436 bits. The same is the case with the rest of the group sizes included in our simulations. We should also observe that *CCost* (*aux*) increases as the network size grows, until the network reaches some threshold value. Then, the corresponding metric starts decreasing. The reason behind this behavior is the following: two members stay connected until the hop distance between them reaches a certain threshold. After this threshold is exceeded, the members are considered disconnected. As the network size increases, the density of members decreases, and

naturally the "neighbors" of each member decrease, and *CCost*(*aux*) and *CompCost*(*aux*) metrics become lower.

In addition, we also verify that the metrics associated with the auxiliary framework are kept reasonably low and add little to the overall overhead produced by the execution of the *wt*-versions, even under a growing secure group and/or network size. On the other hand, the control messages used for the generation of the framework have size $K_S << K$. Acceptable bit lengths for K are above 2048 bits so that a KM protocol can be considered computationally secure to-date. We illustrate the above with an indicative arithmetic example: We have found that *avg* (*CCost* (ING_Opt, n=32, S=300) = 3,925, *avg* (*CCost* (ING, n=32, S = 300) = 5,976, and *avg* (*CCost* (*aux*, ING, n = 32, S = 300) = 395,881. It can be seen that the larger the constant K becomes, the bigger the difference in the overhead of ING_Opt *vs.* ING becomes. However, even if we assume that the bit size of K is as small as 1024 bits, we obtain the following results: *CCost* (ING_Opt + *aux*) = 3925×1024 + 395,881 = 4,019,200 + 395,881 = 4,415,081 bits, while *CCost* (ING) = 5976×1024 = 6,619,424. Obviously, the difference in the overall overhead (including the auxiliary overhead for the case of *wt*-ING) is considerable, even under this worst case scenario (small K, highest observed *CCost*(*aux*)). The difference in the overall overhead becomes even more impressive if GDH.1 is considered. We emphasize again that a new framework needs not be computed every time a *wt*-KA protocol is executed. We can re-compute the framework periodically. Hence, the impact of the framework overhead on the overall cost of our algorithms is even lower in practice.

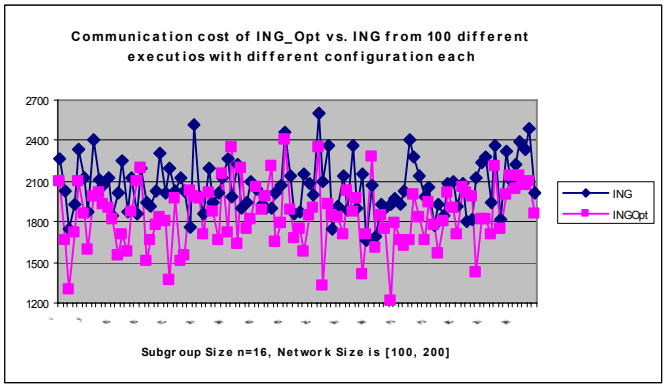


Figure 1. *CCost* of ING_Opt *vs*. ING, for group size n = 16, and network size S = 100, for 100 different graph configurations. ING_Opt results in substantially superior performance for the vast majority of graph configurations.

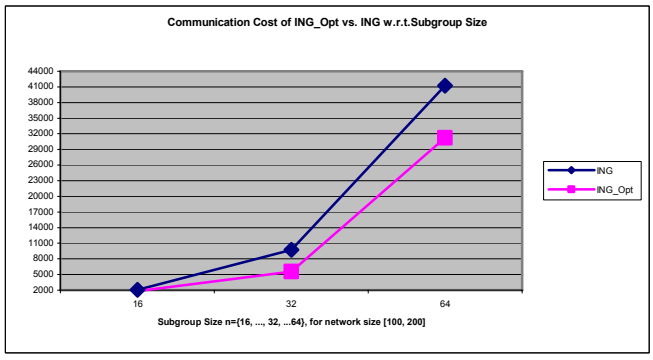
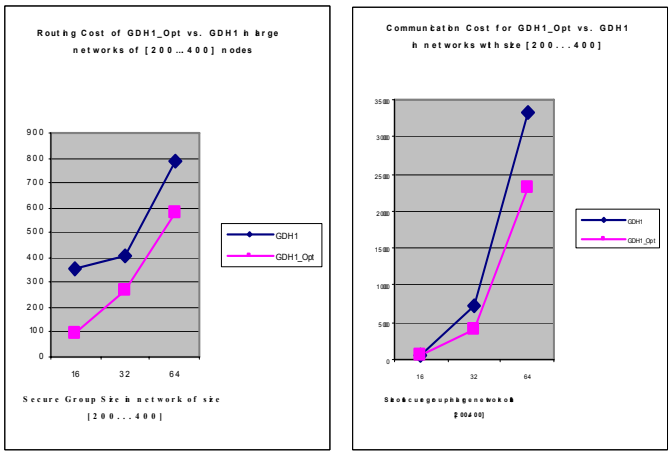


Figure 2. *CCost* of ING_Opt *vs*. ING, for group size <16,…, 64>, in a network of size <100, 200>. ING_Opt demonstrates considerably superior performance.



Figures 3, 4. *RCost* (3) and *CCost* (4) of GDH.1_Opt *vs.* GDH.1 w.r.t. number of group members [16, 32, 64] in a large network of [200,…, 400] nodes.

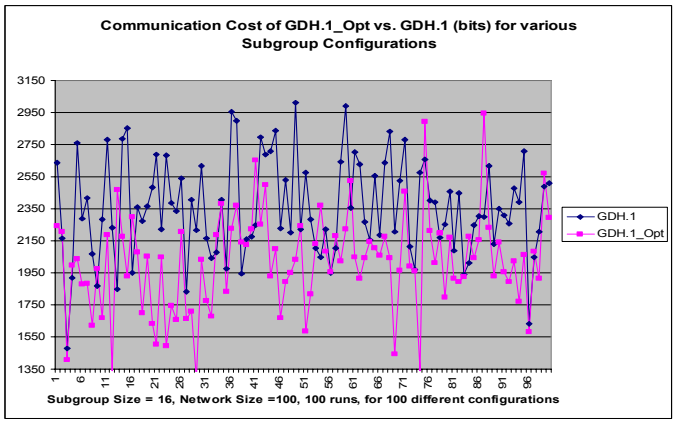
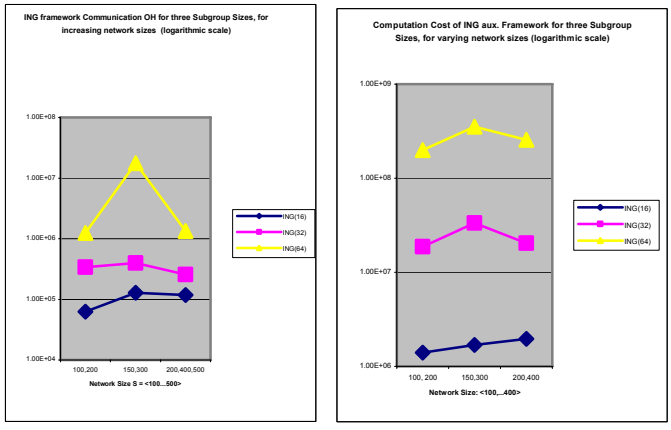


Figure 5. *CCost* of GDH.1_Opt *vs.* GDH.1 for 100 different subgroup configurations on a network of 100 nodes (100 different runs) for a subgroup size of 16 members. The performance of our algorithm is significantly superior in the vast majority of cases, and this reflects on the average case as well. The ratio of improvement in these cases becomes: $0.53 < R_{COMM} < 0.70$.



Figures 6, 7. *CCost*(*aux*) (bits) [6], and *CompCost*(*aux*) (bits) [7] for ING_Opt, for an increasing group size <16,…,64>, for three different scenarios of network size: S1=<100,200>, S2 = < 150, 400>, S3 = <200, 500>. The corresponding costs increase with the group size. They also increase with the network size up to a certain threshold, as discussed.

Note: The behavior and performance of GDH.2 is similar to this of GDH.1, and we skip the related results for lack of space.

VIII. CONCLUSION

This paper focuses on the design and analysis of topology-oriented versions of a number of the following DH-based KA protocols over wire-less multi-hop ad hoc networks: GDH.1, GDH.2, ING. We describe new methods for approximating (and potentially optimizing) the most significant metrics of interest associated with these protocols: the communication, routing, and latency functions. We introduce heuristics for generating topology-oriented communication schedules, on top of which the discussed protocols are executed. The algorithms introduced address generalized secure member graphs. On a generalized secure member graph, the notion of connectivity has been relaxed, i.e. the communication path between two members may as well include non-member relays, as long as the number of hops in the path is lower than a given threshold. The algorithms introduced in our previous work [23] were designed for fully connected secure member graphs. This specialized model however does not effectively capture a realistic, general multi-hop ad hoc network. Applying the methods suggested in [23] on the generalized graphs of our current study makes no sense in most cases or results in a substantial deterioration of the expected performance of the protocols. Hence, in this paper we re-designed the communication schedules of the given protocols by proposing new techniques that address a totally generic network. The heuristics introduced achieve much better approximations of the metric functions we want to optimize. Our comparisons of the new topology oriented and the original KA schemes are done via simulations. Indeed, the new protocols achieve a significant reduction in the communication and routing overhead. We believe that there is scope for even more efficient topology oriented approaches and we consider efficient simulations of KA protocols over multi-hop ad hoc networks as an interesting open problem.

REFERENCES

[1] K. Becker, U. Wille, "Communication Complexity of Group Key Distribution," *Proc.5th ACM Conference on Computer & Communicatios Security*, pp. 1-6, San Francisco, CA, November 1998.

[2] M. Steiner, G. Tsudik, M. Waidner, "Diffie-Hellman Key Distribution Extended to Groups," *3rd ACM Conference on Computer & Communication Security*, pp. 31-37 ACM Press, 1996.

[3] W.Diffie, M.Hellman,"New directions in cryptography", *IEEE Trans. on Information Theory*, 22(1976), 644-654.

[4] I. Ingemarsson, D.Tang, C.Wong. "A Conference Key Distribution System", *IEEE Trans. on Information Theory*, 28(5): 714-720, Sept. 1982

[5] M.Burmester, Y.Desmedt. "A Secure and Efficient Conference Key Distribution System", *Advances in Cryptology–EUROCRYPT'94, Lecture Notes in Computer Science.* Springer – Verlag, Berlin, Germany.

[6] M. Hietalahti. "Key Establishment in Ad-Hoc Networks," *M.S. Thesis*, Helsinki University of Technology, Dept. of Computer Science and Engineering, May 2001.

[7] A.Perrig, "Efficient Collaborative Key Management Protocols for Secure Autonomous Group Communication," *Int'l Workshop on Cryptographic Techniques and E-Commerce (CrypTEC'99),* pp. 192-202, July 1999.

[8] N.Asokan, P. Ginzboorg, "Key-Agreement in Ad-Hoc Networks," *Computer Communications*, Vol. 23, number 18, pp. 1627-1637, 2000.

[9] Y. Kim, A. Perrig, G. Tsudik, "Simple and Fault Tolerant Key Agreement for Dynamic Collaborative Groups," *Proc. 7th ACM Conf. on Computer and Communication Security (CCS 2000),* pp. 235-244.

[10] J. Katz, M.Yung, " Scalable Protocols for Authenticated Key Exchange", *Advances in Cryptology - EUROCRYPT'03, Springer-Verlag*, LNCS Vol 2729, pp. 110-125, Santa Barbara, USA.

[11] J.Katz, R.Ostrovski, A.Smith, "Round Efficiency of Multi-Party Computation with a Dishonest Majority", *Advances in Cryptology, EUROCRYPT'03*, LNCS Vol. 3152, pp.578-595, Santa Barbara, USA.

[12] Y.Amir, Y.Kim, C.Rotaru, J.Schultz, G.Tsudik, "Exploring Robustness in Group Key Agreement", *Proc. of the 21th IEEE Int'l Conference on Distr. Computing Systems,* pp. 399-408, Phoenix, AZ, April 16-19, 2001.

[13] Y.Amir, Y.Kim, C.Rotaru, J.Schultz, J.Stanton, G.Tsudik, "Secure Group Comm/tion using Robust Contributory Key Agreement"., *IEEE Trans. on Parallel and Distributed Systems*, Vol. 15, number 5, pp. 468-480, May '04.

[14] L.Zhou, Z.Haas, "Securing Adhoc Networks," *IEEE Network Magazine*, vol. 13, number.6, pp. 24-30, Nov/Dec 1999.

[15] J.Kong, P.Zerfos, H.Luo, S.Lu, L.Zhang, "Providing Robust and Ubiquitous Security Support for Wireless Ad-Hoc Networks," *Proc. 2001 IEEE Int'l Conf. on Network Protocols (ICNP 2001)*, pp. 251-260.

[16] S.Capkun, L.Buttyan, J.Hubaux, "Self-Organized Public Key Management for MANET," *IEEE Trans. on Mobile Computing*, Vol. 2, number 1, pp. 52-64, Jan-Mar. 2003.

[17] L.Eschenauer, V.Gligor., "A Key Management Scheme for Distributed Sensor Networks," *Proc. 9th ACM Conference on Computer and Communication Security (CCS'02)*, pp. 41-47, Nov, 2002.

[18] H.Chan, A.Perrig, D.Song, "Random Key Predistribution Schemes for Sensor Networks," *Proc. 2003 IEEE Symposium on Security and Privacy*, pp. 197-213, May 2003.

[19] S.Yi, R.Kravets, "Key Management for Heterogeneous Ad hoc Wireless Networks," University of Illinois, Urbana-Champaign, CS dept., TR#UIUCDCS-R-2001-2241, UILU-ENG-2001-1748, July 2002.

[20] L.Lazos, R. Poovendran, "Energy-aware Secure Multicast Comm/tion in Ad-hoc networks Using Geographic Location Information", *IEEE Int'l Conf. of Acoustic Speech Signal Processing (ICASSP'03),* pp. 201-204, Hong Kong, China, April, 2003.

[21] S.Zhu, S.Setia, S.Xu, S.Jajodia, "GKMPAN: An Efficient Group Re-keying Scheme for Secure Multicast in Ad-hoc Networks", *IEEE Computer Society, MobiQuitous 2004,* pp. 45-51.

[22] M.Striki, J.Baras "*Efficient Scalable Key Agreement Protocols for Secure Multicast Communication in MANETs*", Collaborative Technologies Alliance (CTA) Symposium, College Park MD, May 2003.

[23] M.Striki, J.Baras, G.DiCrescenzo, "Modeling Key Agreement Protocols in Multi-Hop Ad Hoc Networks", *Proc. 2006 Int'l Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 39-45, Vancouver, CA, July 2006

[24] R. Gallager, P.Humblet, P.Spira. "A distributed algorithm for minimum weight spanning trees", *ACM Transactions on Programming Languages and systems (TOPLAS)*, 5(1): pp.66-77, January 1983

[25] T.Cormen, C.Leiserson, R.Rivest,"Introduction to Algorithms", *MIT [2]Press, McGraw Hill Book Company*, March 1990

[2] The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the Army Research Laboratory or the U.S. Government.