VP3: Using Vertex Path and Power Proximity for Energy Efficient Key Distribution

Loukas Lazos, Javier Salido and Radha Poovendran
Network Security Lab, Dept. of EE,
University of Washington, Seattle, WA 98195-2500
{l_lazos, javiers, radha}@ee.washington.edu

***Abstract*— We study the problem of energy-efficient key distribution for securing multicast communications in wireless ad hoc networks. Recently, we showed that a cross-layer design approach for key distribution, incorporating network layer (routing) as well as physical layer (energy) parameters, leads to energy savings. We also showed that heuristics are needed to reduce the computational complexity. In this paper we show that further reduction in energy expenditure is achieved by assigning common keys to nodes the receive messages from a sender via a common path. We develop a computationally viable heuristic called VP3 that uses codewords to represent paths and groups nodes based on the length of the common path, derived by the *Hamming distance* between codewords. We also present simulation results to illustrate the improvements achieved by VP3.**

I. Introduction

Many network operations such as routing, neighbor discovery or topology control, require identical data to be delivered from a single sender to multiple receivers. In such operations, the multicast communication model reduces the network traffic, thus saving energy resources. In addition, due to the nature of the wireless medium, if several receivers are within the communication range of the sender, a single transmission to the furthest receiver is sufficient to deliver the message to all of them. This property, known as the *broadcast advantage* (BA) [2], leads to significant energy conservation. However, anyone within the communication range has access to transmitted information. In many critical applications such as military networks, information privacy needs to be preserved.

Session encryption can be used to secure the wireless multicast communication. If the session contains large volumes of data, symmetric key cryptography is energy-efficient compared to asymmetric key cryptography even in wired networks. However, the use of symmetric key requires that every valid member of the multicast group has access to the shared symmetric key for decryption, known as *Session Encryption Key* (SEK).

In dynamic multicast groups where members may join or leave the group, the SEK needs to be updated when a membership change occurs, in order to preserve the secrecy of future and past communication and ensure that only the valid members have access to multicast communication at any given time. Hence, the sender must have secure channel(s) to communicate with the valid members of the group at any time. To securely update the SEK, each member needs to possess additional *Key Encrypting Keys* (KEK), that are used to encrypt the SEK. Since the sender must be able to reach all valid members, the SEK update problem reduces to the *key distribution problem*, i.e. finding *efficient* algorithm(s) to distribute the KEKs in order to *efficiently* update the SEK under membership changes.

The multicast key distribution problem in the context of wired networks has been extensively studied [5]–[7]. The key distribution trees, independently proposed in [5], [6] have been adopted as a scalable solution in terms of key update messages sent by the sender, also known as the *Group Controller* (GC), while also maintaining a low storage requirement at the receiver. However, the key tree solutions developed for wired networks were shown to be energy inefficient for wireless ad hoc networks [1], [3], [4]. In [1], we showed that a cross-layer design approach jointly considering the routing layer and the physical layer is important to design energy-efficient key distribution schemes. In [1], we also showed that the computational complexity of the optimal energy-efficient solution is at least $\mathcal{O}(N^4)$, and proposed a heuristic key distribution scheme with complexity $\mathcal{O}\log(N)$. In doing so, we made use of the power proximity of the network nodes with respect to the GC.

Our contributions: In this paper we propose a novel key distribution scheme that achieves energy efficiency by assigning common KEKs to nodes that share the longest common path to the GC. Energy savings occur since any key update encrypted with a KEK known to members sharing a common path will traverse the common path only once and then be dispersed to the intended group members. To identify common paths between nodes we adopt an algebraic approach where paths are represented as codewords, and the length of the common path is derived by computing the Hamming distance between two codewords [8].

The remainder of the paper is organized as follows. In Section II, we present relevant background and prior work. In Section III, we describe the network model assumed. In Section IV, we analyse how the "network direction" impacts the energy efficiency of the key distribution. In Section V we present our key distribution algorithm. In Section VI, we provide simulation results and show the improvements achieved by our algorithms. Section VII presents conclusions.

II. Background & Prior Work

In this section we present background on logical key trees used as a building block to our algorithm, and describe our

prior work on cross-layer design on key distribution [1].

A. Logical key hierarchical trees

While adding a new member, SEK is updated to protect the past traffic. When a member is revoked the SEK and possibly all KEKs known to the deleted member are updated to protect future communication from being exposed to the deleted member. Key tree hierarchies were independently proposed for key distribution in wired networks, in [5], [6], in order to reduce the cost of member deletion. Key trees require $\mathcal{O}\log(N)$ communication overhead and $\mathcal{O}\log(N)$ storage at the member and hence, scale with the group size N.

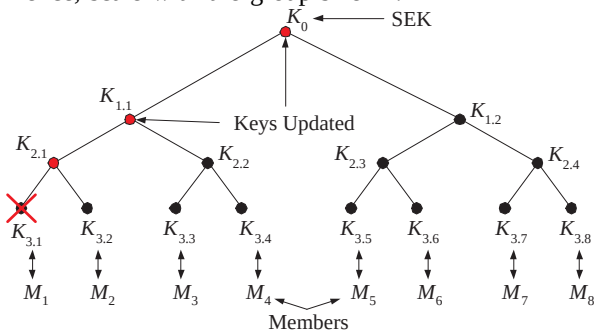


Fig. 1. A binary logical key tree. Members are placed at the leaf nodes. Each member holds the keys traced along the path from the leaf to the root. If M_1 leaves the multicast group keys (K_0, $K_{1.1}$) need to be updated.

In figure 1(a), we present a binary key distribution tree for a multicast group of $N = 8$ members plus the GC. Such a tree is a visual representation of the key assignment to each member of the multicast group. Referring to figure 1(a), each member is associated with a single leaf node of the tree. Each member is assigned keys that are traced along the path from the leaf node to the root [5]. For example, M_1 is assigned keys $\{K_0, K_{1.1}, K_{2.1}, K_{3.1}\}$. K_0 is used as the SEK, while the rest of the keys are KEKs. If M_1 leaves the multicast group, keys $\{K_0, K_{1.1}\}$ need to be updated, while key $K_{2.1}$ need not be updated, since it is not shared by multiple members.

B. Cross-layer design for key distribution

In [1], we showed that logical key hierarchical trees proposed for wired networks are energy inefficient since they do not take into consideration the network topology. We proposed a new metric called *average update energy* E_{Ave} to incorporate the energy parameter in the key tree design. The average update energy is defined as the energy required to update keys if each member were to be deleted from the multicast group:

$$E_{Ave}(R,T) = \frac{1}{N}\sum_{i=1}^{N} E_{M_i}(R,T) \qquad (1)$$

where $E_{M_i}(R,T)$ denotes the energy required to update keys after the deletion of M_i, according to the key distribution tree T and routing tree R.

We also showed that finding the key tree T^* that minimizes (1), requires an algorithm of complexity at least $\mathcal{O}(N^4)$, and proposed *RAwKey* that constructs an energy-efficient key tree with a complexity of $\mathcal{O}\log(N)$, based on the power proximity of members with the GC.

The RAwKey consisted of the following steps: (a) compute the network energy expenditure E_i required to reach each member from the GC, (b) sort the members according to E_i in ascending order and assign each member to a unique leaf node of the tree. By grouping members (assigning common KEKs) that have the smallest energy difference, in many cases, we group nodes that receive messages through common routing paths and hence save energy resources. However, as we will show in section IV this need not be always true. We then show that we can improve upon the energy efficiency at the expense of algorithmic complexity. We first present our network model assumptions.

III. Network Model Assumptions

Network generation: We assume that the network consists of N multicast members plus the GC, randomly distributed in a specific area. We consider a single-sender multiple-receiver communication model. All users are capable of corroboratively relaying information between an origin and destination. We also assume that nodes have the ability to generate and manage cryptographic keys.

Network initialization: We assume that the network is successfully initialized and initial cryptographic quantities (at least pairwise trust) have been distributed. Several novel approaches that address the critical problem of secure initialization in ad hoc networks have been presented in [9], [10].

Layer Interaction: We further assume that information from the network layer such as routing paths, and information from the physical layer such as transmission power is available at the application layer where the key distribution algorithm is executed.

Since our goal in this paper is to design key management algorithms and not protocols, we do not address the MAC layer implementation requirements of our algorithms.

IV. Forming Groups from Common Paths

In this section we show how energy savings occur when common keys are assigned to members that share common paths. Intuitively, if a message is sent to several members that, according to the routing tree receive information through the same path, the message will have to traverse the common path only once and hence save energy resources.

In figure 2(a) we show the routing tree of an ad hoc network with $\{M_1 \sim M_4\}$ being the members of the multicast group. The energy units (E.U.) to maintain each link are also shown. In figure 2(b) we show key Tree A for the multicast group of figure 2(a), constructed according to our key distribution scheme in [1]. The energies E_i required to reach M_i from the GC are sorted in ascending order $\{E_1 < E_3 < E_2 < E_4\}$ and the members are placed in the same order in the leafs of the key tree, from left to right.

In figure 2(b), the assignment of a common key $K_{1.1}$ to $\{M_1, M_3\}$ leads to energy savings since for example updating K_0 with $K_{1.1}$ requires the transmission of just one message to

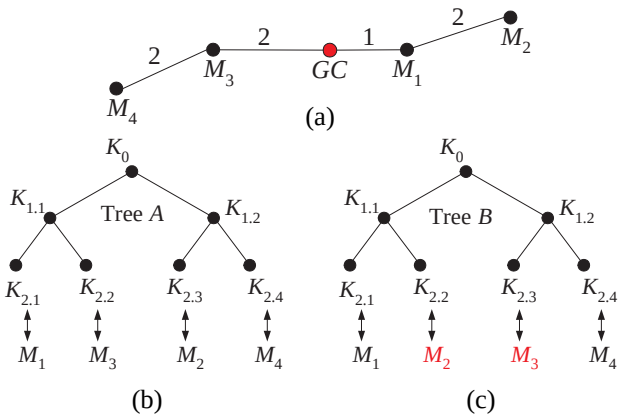


Fig. 2. (a) A multicast network with four members and a GC. Each link energies are indicated. (b) A binary, power-proximity key tree. Members are placed at the leaf nodes from left to right in ascending order of energies to reach them from the GC. (c) A binary, routing-aware tree that combines the power proximity and the maximal common routing path to group nodes.

member M_3. However, the assignment of a common key $K_{1.2}$ to $\{M_2, M_4\}$ is not efficient since $\{M_2, M_4\}$ are in opposite "network directions." Hence *power proximity* is not sufficient to group members together.

In figure 2(c), we present key Tree B for the multicast group in figure 2(a) that takes into account the routing paths. Members are placed adjacently at the leaves of the key tree if the share the longest common path to GC. Hence, M_1 is grouped with M_2 and M_3 with M_4. By computing the $E^A_{Ave}(R,T)$ and $E^B_{Ave}(R,T)$, for trees A, B respectively, we get $E^A_{Ave}(R,T) = 26$ E.U., while $E^B_{Ave}(R,T) = 24$ E.U.

The difference in the energy consumption comes for the fact that in Tree B, key updates destined to a group of members are routed only through members of that group. On the other hand, in key Tree A a key update destined to $\{M_2, M_4\}$ is routed through nodes M_1 and M_3 thus wasting energy.

Based on our observation that grouping members in the key distribution tree according to the length of the common path to the GC leads to energy conservation, we propose a key distribution scheme called VP3.

V. VP3: Vertex-Path Power Proximity Algorithm

A. Idea of VP3

Our vertex-path power proximity algorithm uses a metric proposed for network tomography in [8]. The author suggests the representation of each path as a binary codeword of length equal to the network size, with "ones" corresponding to the nodes traversed by the path and zero otherwise. The measure of difference between any two paths is equal to the *hamming distance* H_d between the two codewords corresponding to those paths [8].

By representing the paths from the GC to every member of the multicast group as codewords and computing the hamming distances between the codewords, we can compute the length of the common path form the GC to any group of members. Then we group members that share the longest common paths. A detailed description of our algorithm is given below:

B. Detailed description of VP3

Our algorithm assumes two sets of parameters as inputs: (a) the NxN connectivity matrix C, where each row C_i is a codeword that represents the node path from the GC to node i, such that the entry $C_{i,j} = 1$ if node j is traversed on the path from GC to i and $i \neq j$, and $C_{i,j} = 0$ otherwise and, (b) a vector E of length N, where the i^{th} entry E_i, indicates the energy expenditure required to transmit a message from GC to node i, following the path indicated by the connectivity matrix C. The steps we need to follow in order to construct an α-ary key distribution tree where α is the degree of the tree, are:

Step 1: Calculate the hamming weight $H_w(i)$ for each row in C, corresponding to the path from the GC to node i. The $H_w(i)$ of row C_i is equal to the number of nodes traversed on the path from GC to node i, not counting i, given by:

$$H_w(i) = \sum_{j=1; j\neq i}^{N} C_{i,j} \tag{2}$$

Step 2: Choose the node i^* with the *largest hamming weight* $H_w(i^*) = \max(H_w(i))$. If there is more than one node with the largest H_w, then pick i^* to be the one requiring the maximum energy E_{i^*} to be reached from GC, out of all the nodes with the largest hamming weight. If more that $(\alpha - 1)$ nodes have the smallest $H_d(i,j)$ and maximum E_i, pick $(\alpha - 1)$ randomly.

Step 3: Pick the $(\alpha - 1)$ nodes with the smallest hamming distances H_d from i^*. The hamming distance $H_d(i,j)$ between two nodes i, j is calculated by executing a bitwise exclusive OR operation (XOR) between the codewords corresponding to the nodes i, j and adding the bits of the resulting codeword.

$$H_d(i,j) = \sum C_i \otimes C_j \tag{3}$$

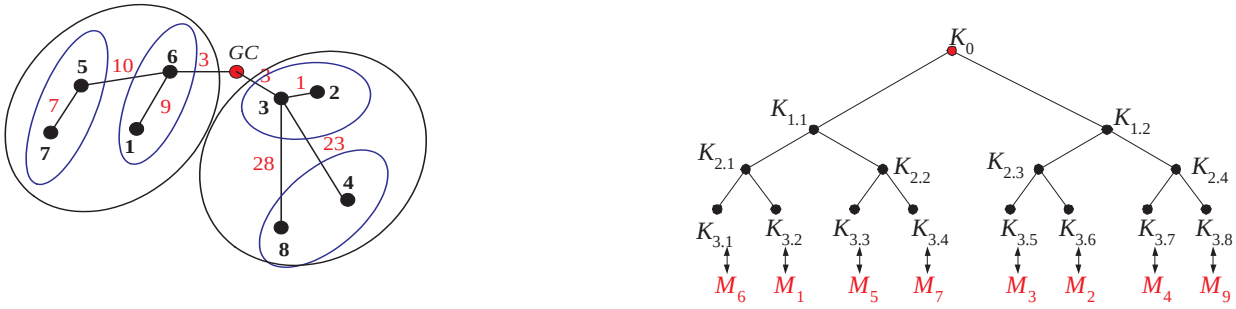
If there are more than $(\alpha - 1)$ nodes with equal H_d to i^* always pick, if any, the node found in the path from the GC to i^*. For the remaining $(\alpha - 2)$ nodes, pick the ones with the largest E_i. If more that $(\alpha - 2)$ nodes have the smallest $H_d(i,j)$ and maximum E_i, pick $(\alpha - 2)$ randomly. Assign a unique common key to all members corresponding to the α nodes chosen in this step.

Step 4: Repeat Steps 2,3 until all nodes belong in clusters of at most α nodes and are assigned a unique common key.

Step 5: Generate a matrix C' with rows corresponding to the clusters generated in Step 4 and columns corresponding to the network nodes. An entry $C'_{i,j} = 1$ if node j is traversed in the path from the GC to any of the members of cluster i, and $C'_{i,j} = 0$ otherwise. Compute the vector E' where the i^{th} entry indicates the energy expenditure required to transmit a message from GC to all members of cluster i, following the paths indicated by the connectivity matrix C'. Execute Steps $1 \sim 4$ with inputs C', E', instead of C and E.

Step 6: Repeat Steps $1 \sim 5$ until all nodes belong to a single cluster.

In figure 4 we present the pseudo-code for VP3. The $ConnectivityMatrix()$ function computes the connectivity matrix for its argument set. The $EnergyMatrix()$ function



(a) (b)

Fig. 3. (a) The broadcast routing tree for an ad hoc network of eight nodes plus the GC. Nodes $\{1 \sim 8\}$ are multicast group members. The numbers on the links indicate the energy expenditure to transmit a message through that link. The circles indicate the grouping of the members into the key tree after the execution of VP3. (b) The key distribution tree constructed with VP3.

VP3: Vertex-Path Power Proximity Algorithm

$MG =$ Multicast group
$C = ConnectivityMatrix(MG)$
$E = EnergyMatrix(MG)$
for $l = 1 : \lceil \log_\alpha(N) \rceil$
  $H_w(i) = \sum_{j=1;j\neq i}^{N} C_{i,j}, \forall$ rows C_i
  for $k = 1 : \lceil \frac{N}{\alpha^l} \rceil$
    $i^* = \arg\max_{i \in MG} H_w(i)$
    if $|i^*| > 1$
    $i^* = \arg\max_{i \in i^*} E_i, \quad MG = MG \backslash \{i^*\}$
    $j' = \{j \in MG \ \ni \ \arg\min_{j \in MG} H_d(i^*, j)\}$
    If $|j'| > (\alpha - 1)$ pick j' path $GC \rightarrow i^*$
      and $(\alpha - 2) \in j' \ni \arg\max_{i \in j'} E_i$
    $MG = MG \backslash \{j'\}, \quad G = G \cup j'$
    $AssignKey(j')$
  endfor
  $MG = G$
  $C = ConnectivityMatrix(MG)$
  $E = EnergyMatrix(MG)$
endfor

Fig. 4. Pseudo-code for VP3. The $ConnectivityMatrix()$ function computes the connectivity matrix for its argument set. The $EnergyMatrix()$ function computes the energy required to reach a group of vertices from the GC, where the groups are elements of the vector argument. The $AssignKey$ function assigns a common key to every element of the argument set.

computes the energy required to reach a set of nodes sharing a common key from the GC, where each set is an element of the argument. Initially, the argument to both functions is the set of all members of the multicast group MG. With the construction of every subsequent level l of the key tree, the argument will be the set of groups generated in the previous level. The $AssignKey$ function assigns a common key to every element of the argument set.

C. Algorithmic complexity of VP3

In the worst case of a binary key tree construction, the algorithmic complexity of VP3 is $\mathcal{O}(N^2)$, which is higher than the $\mathcal{O}(\log(N))$ complexity of heuristics in [1]. However, as we will show in section VI, VP3 achieves significant reduction in the $E_{Ave}(R,T)$ over the best known algorithm in [1].

D. A walk through VP3

In this section we present an application of VP3 on a sample network. Consider figure 3(a), where we show an ad hoc network and its broadcast routing topology. The numbers on the links indicate the energy link cost. Nodes $1 \sim 8$ correspond to members $M_1 \sim M_8$ of the multicast group MG. Table I shows the connectivity matrix C for MG, the hamming weights $H_w(i)$ for each row C_i, and the energy expenditure E_i necessary to reach member M_i from the GC.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	H_w	E
1	0	0	0	0	0	1	0	0	1	12
2	0	0	1	0	0	0	0	0	1	4
3	0	0	0	0	0	0	0	0	0	3
4	0	0	1	0	0	0	0	0	1	24
5	0	0	0	0	0	1	0	0	1	13
6	0	0	0	0	0	0	0	0	0	3
7	0	0	0	0	1	1	0	0	2	20
8	0	0	1	0	0	0	0	0	1	29

TABLE I
CONNECTIVITY MATRIX, HAMMING WEIGHTS AND ENERGY EXPENDITURE FOR THE MULTICAST GROUP IN FIGURE 3(A).

We want to construct a binary key tree ($\alpha = 2$) using VP3. Column H_w in Table I shows the result of executing Step 1. In Step 2, we identify the set of paths with the greatest H_w. In our example, we select 7, that has the maximum H_w and withdraw it from the pool.

In Step 3, we find nodes $\{1, 5\}$ to have the shortest hamming distance to 7. Since we need to choose only one node $(\alpha = 2)$ and 5 is on the path from GC to 7, $\{M_5, M_7\}$ are assigned a unique common key, and node 5 is also withdrawn from the pool.

In Step 4, we repeat Steps 2,3. In Step 2, nodes $\{1, 2, 4, 8\}$ have the highest Hamming weight, and 8 is selected since it has the highest E_i. In Step 3, nodes $\{4, 2\}$ have the smallest H_d to 8, but none of them is in the path from the GC to 8. Hence, we pair 8 with 4 since 4 has a greater E_i. Members $\{M_8, M_4\}$ are

assigned a unique common key, and nodes $\{4, 8\}$ are withdrawn from the pool. Steps 2,3 are repeated with $\{M_2, M_3\}$ and $\{M_1, M_6\}$ also being assigned a unique common key.

In Step 5, we recompute the connectivity matrix C' and energy matrix E' for the pairs generated in Step 4, and repeat Steps 1 to 4. Nodes $\{2, 3, 4, 8\}$, $\{1, 5, 6, 7\}$ are grouped and members $\{M_2, M_3, M_4, M_8\}$, $\{M_1, M_5, M_6, M_7\}$, are assigned a unique common key respectively. At this point the SEK is assigned to all members and the key tree construction is completed. Figure 3(b) shows the resulting key distribution tree.

VI. Simulation Comparison

While we already presented a counter example in figure 2 showing that the RAwKey in [1] can be improved, using the path information, we present the simulations for completion.

We generated random network topologies confined in a region of size 10x10. Following the network generation, we use the Broadcast Incremental Power (BIP) algorithm [2] to construct and acquire the routing paths from the GC to every group member (Any other suitable routing algorithm can be applied as well). The routing tree was used as an input to the VP3 and was also used to calculate $E_{Ave}(R, T)$.

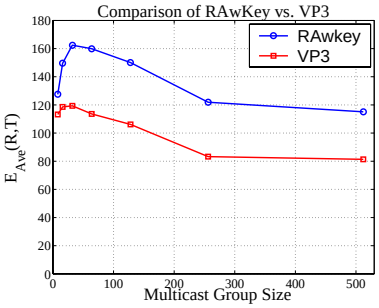


Fig. 5. Comparison of VP3 with RAwkey for different group sizes.

In figure 5, we compare VP3 with RAwKey in [1] for different multicast group sizes. We observe that VP3 requires less average update energy $E_{Ave}(R, T)$ than RAwKey to perform key updates, at the expense of increased algorithmic complexity. Note that the increase in the multicast group size does not increase the $E_{Ave}(R, T)$, for groups bigger than 32 nodes in our simulation. This is due to the fact that since the deployment region was kept constant, as the node density increased more efficient paths were used to deliver key updates to group members, eliminating the high-power transmission links. Therefore, in figure 5, $E_{Ave}(R, T)$ decreases as the network size increases. Also, note that $E_{Ave}(R, T)$ increases as the group size increases from 8 to 16 to 32, since the increase in number of key updates is not compensated by more efficient route discovery. Due to page limit, the analytical evaluation of the efficiency is left for a future, detailed paper.

In figure 6, we show the reduction in energy expenditure achieved by VP3 over RAwKey. We observe that VP3 can yield a reduction in energy expenditure compared to RAwKey up to 31% while the average reduction achieved is 25%. The consideration of the "network direction" through the discovery

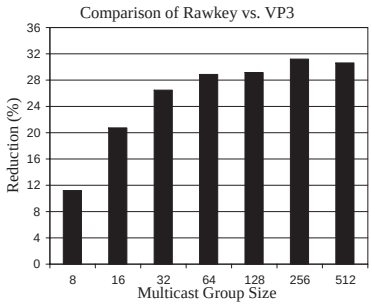


Fig. 6. Reduction in $E_{Ave}(R, T)$ achieved by VP3, over RAwkey.

of nodes sharing common paths and the assignment of common keys to such nodes, led to significant energy conservation for key update operations.

VII. Conclusions

In this paper, we presented currently best performing cross-layer design algorithm for multicast key distribution that uses routing energies from the sender and Hamming codes representing the paths from the sender to each node to minimize the average energy for key updates. Many interesting questions including the performance evaluation remain open in this new problem.

Acknowledgements

This work was supported by Collaborative Technology Alliance for Communication & Networks sponsored by the U.S. Army Laboratory under Cooperative Agreement DAAD19-01-2-0011 and National Science Foundation grant ANI-0093187 and ARO grant DAAD19-0210242. Any opinions findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, Army Research Office or the Army Research Laboratory of the U.S. Government.

References

[1] L. Lazos and R. Poovendran, Cross-Layer Design for Energy-Efficient Secure Multicast Communications in Ad Hoc Networks, In *Proc. of IEEE ICC 2004*, Paris, France, May 2004.

[2] J.E. Wieselthier, G.D. Nguyen and A. Ephremides, On the Construction of Energy Efficient Broadcast and Multicast Trees in Wireless Networks, In *Proc. of INFOCOM 2000*, March 2000.

[3] L. Lazos and R. Poovendran, Secure Broadcast in Energy-Aware Wireless Sensor Networks, in *Proc. of IEEE ISWC 2002*, Victoria, Sep. 2002.

[4] L. Lazos and R. Poovendran, Energy-Aware Secure Multicast Communication in Ad-hoc Networks Using Geographic Location Information, In Proc of IEEE ICASSP 2003, April 2003, Hong Kong, China.

[5] D.M. Wallner, E.C. Harder and R.C. Agee, Key Management for Multicast: Issues and Architectures, INTERNET DRAFT, Sep. 1998.

[6] C.K. Wong, M. Gouda and S. Lam, Secure Group Communications Using Key Graphs, In IEEE/ACM Transactions on Networking, Feb. 2000 vol. 8, no.1, pp. 16-31.

[7] Y. Yang, X. Li and S. Lam, Reliable Group Rekeying: Design and Performance Analysis, In *Proc. of SIGCOMM 2001*, August 2001.

[8] Y. Vardi, Metrics Useful in Network Tomography Studies, In IEEE Signal Processing Letters, Vol 11, Num. 3, pp. 353-355, March 2004.

[9] D.W. Carman G.H. Cirincione and B.J. Matt, Energy-Efficient and Low-Latency Key Management for Sensor Networks, In *Proc. of 23rd Army Science Conference*, December 2002.

[10] F. Stajano and R. Anderson, The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks, IN *Proc. of SPW 1999*, April 1999.