Energy and Bandwidth-Efficient Key Distribution in Wireless Ad-Hoc Networks: A Cross-Layer Approach

Javier Salido, Loukas Lazos, and Radha Poovendran †
Network Security Lab (NSL), Seattle, WA

***Abstract*— We address the problem of resource-efficient access control for group communications in wireless ad-hoc networks. Restricting the access to group data can be reduced to the problem of securely distributing cryptographic keys to group members, known as the key distribution problem (KDP). We examine the KDP under four metrics: (a) member key storage, (b) group controller (GC) transmissions, (c) multicast group (MG) update messages, and (d) average update energy. For each metric, we formulate an optimization problem and show that the KDP has unique solutions for metrics (a) and (b), while is NP-complete for (c) and (d). We propose a cross-layer heuristic algorithm called VP3 that bounds member key storage, and GC transmissions, while significantly reducing the energy and bandwidth consumption of the network. We define the notion of path divergence as a measure of bandwidth efficiency of multicasting, and establish an analytical worst-case bound for it. Finally, we propose On-line VP3 which dynamically updates the key assignment structure according to the dynamics of the communication group in a resource-efficient way.**

***Index Terms*— key distribution, secure group communication, key management, security, multicast, ad-hoc.**

I. Introduction

Wireless ad-hoc networks are envisioned to provide rapid on-demand network deployment due to their self-configurability and lack of pre-deployed infrastructure requirements. Since ad-hoc networks rely on the collaboration principle, a significant amount of communication is needed to provide any service. However, if nodes are equipped with omnidirectional antennae, a single transmission suffices to deliver the same message to any receiver within communication range. Hence, protocols employing *Group Communications* can benefit from the broadcast nature of the wireless medium, and reduce the energy expended for communication [1], [24].

While broadcasting in group communications provides both energy and bandwidth efficiency, unencrypted transmissions are openly available to any eavesdropper. Hence, *Access Control* policies are necessary in order to restrict access to the contents of multicast transmissions to valid members of the *Multicast Group* (MG). One approach to control access to the group communication is via broadcast encryption [6]. Alternatively, a bandwidth and computationally efficient solution to this problem uses a single symmetric cryptographic key, called the *Session Encryption Key* (SEK), that is shared by the multicast source and all members of MG [2], [23], [25]. Using the SEK, the sender needs to perform only one encryption and one transmission to send data to MG, while MG members need only perform a single decryption to receive the data.

In the case where the multicast group MG is dynamic, the valid members of MG need to be updated with a new SEK after every membership change so that new members do not access past data (backward secrecy [2], [23], [25]), and departing members do not access future transmissions (forward secrecy [2], [23], [25]). In order to update the SEK, additional keys called *Key Encryption Keys* (KEKs) are used by the entity managing the cryptographic keys, known as the *Group Controller* (GC). Hence, the problem of controlling access to the multicast data reduces to the problem of managing and distributing the SEK and KEKs to the members of MG. This problem is known as the *Key Management Problem* or *Key Distribution Problem* (KDP) [2], [23], [25].

Previous research on the KDP in wired networks [2], [23], [25] mainly focused on designing scalable systems that reduce costs in terms of key storage at each member, and number of messages the GC has to transmit to update keys after a membership change. Through the use of tree-based key structures, member key storage and GC transmissions have been reduced to the order of $\mathcal{O}(\log |MG|)$ [2], [23], [25].

While key storage and sender communication cost are important performance metrics even in wireless ad-hoc networks, total energy expended by the network, and total communication overhead, are critical parameters for the viability and operability of many network services, including the secure multicast service, when the network devices are resource limited. However, the energy and total communication overhead were not a major concern in wired networks. Thus, the solutions proposed for the KDP in wired networks [2], [23], [25], are not sufficient for wireless ad-hoc networks.

We note that, for wireless networks, the energy expenditure (physical layer) required for multicasting messages is dependent on the network topology (network layer), and hence, energy-efficient key management (application layer) requires a cross-layer design that makes explicit use of the routing paths. Recent efforts have shown that incorporating the network topology in the key management leads to energy-aware solutions to the KDP problem [11], [13], [21].

A. Our Contributions

We examine the KDP under four metrics, each of which involves optimizing one of four network resources: (a) *member key storage*, (b) GC *transmissions*, (c) number of messages

†This work was supported in part by the following grants: ONR YIP award, N00014-04-1-0479, ARO PECASE grant, W911NF-05-1-0491, and ARL CTA Grant DAAD 19-01-2-0011.

sent by the network to update the SEK and related KEKs, which we refer to as MG *update messages*, and (d) the energy expended by the network for delivering the update messages to valid members of MG after a member deletion, which we refer to as *update energy cost*. We formulate an optimization problem for each metric, and provide the optimal solution when possible. We show that metrics (a) and (b) do not depend on the network topology and unique solutions to the KDP can be obtained that are equivalent to the optimal solutions provided for wired networks [2], [23], [25].

We prove that finding the key assignment structure that minimizes the MG update messages is an NP-complete problem. We further prove that finding the key assignment structure that minimizes update energy cost for rekeying is also an NP-complete problem. To build an energy and bandwidth efficient key assignment structure, a heuristic algorithm called VP3 is proposed. We establish performance bounds for VP3 and through extensive simulations, show that VP3 makes near optimal key assignment decisions. Finally, an $\mathcal{O}(|MG|)$ complexity algorithm is proposed, that performs dynamic maintenance of the key assignment structure, by inserting and deleting members without having to rebuild the key assignment structure after each membership change.

B. Organization of the Paper

Section II presents the notation and formulation of the KDP in wireless ad-hoc networks. Section III analyzes the problem complexity. Section IV introduces our cross-layer algorithm, called VP3, and analyzes its performance bounds. Section V presents On-line VP3, an extension for dynamic maintenance of the key assignment structure. We present our simulation results in Section VI, and our conclusions in Section VII.

II. The Key Distribution Problem in Wireless Ad-Hoc Networks

A. Network Assumptions and Notation

We assume the network consists of N multicast group members plus the GC, randomly distributed in a specific area. We assume the broadcast routing topology R, is known [1], [24], and consider a single-sender multiple-receiver communications model in which all nodes, including the GC are using omnidirectional antennae. All nodes are capable of collaborative relay of information between an origin and a destination. We also assume that nodes have the ability to generate and manage cryptographic keys, and that the length of each key is equal to the length of a single message.

We further assume that pairwise trust has been established between the GC and all group members. This can be achieved by pre-loading a unique pairwise key to every member of the MG, and to the GC. Alternatively, an online Key Distribution Center (KDC) can act as a trusted third party between every member of MG and the GC [17]. Pairwise trust can also be established in the absence of a (KDC), using storage-efficient probabilistic approaches [5]. Finally, we assume the GC has enough storage and energy resources to complete the functions assigned to it. Table I presents the notation we will be using throughout the rest of the paper.

TABLE I
Notation.

Symbol	*Description*
GC	Group Controller.
MG	Multicast Group.
$N = \|MG\|$	Multicast Group size.
M_i	i^{th} member of the multicast group.
T	Key distribution tree.
h	Height of a key distribution tree.
d	Degree of a key distribution tree.
l	Level of a node in the key distribution tree.
$K_{l,j}$	Key assigned to the j^{th} node, at level l in T.
R	The multicast routing tree with set of nodes MG, and set of links A, of a wireless ad-hoc network.
$\{m\}_{K_{l,j}}$	Message m is encrypted with key $K_{l,j}$.
$S_{l,j}(T)$	Set of multicast group members that hold key $K_{l,j}$ in T.
P_{M_i}	Total power required to unicast a message from GC to M_i.
E_{M_i}	Total energy required to unicast a message from GC to M_i.
$E_{M_i \rightarrow M_j}$	Energy expenditure of M_i when transmitting a message to M_j.
E_S	Energy expenditure of the GC and MG, when multicasting to group S.
$A \rightarrow B : m$	A sends message m to B.

B. Adversarial Model

We assume that any set of members $S \subset MG$ may collude in an attempt to construct the set of keys held by a valid member $M_i \notin S$. Construction of the set of keys of a valid member $M_i \notin S$, allows any colluding member to access future key updates for M_i and, hence, access to future communications even if the entire set S is deleted from MG.

For an adversary that is not a member of MG, we assume it can eavesdrop all encrypted communications between participants, but has no access to the cryptographic keys. We do not consider Denial of Service attacks that would prevent a member from receiving keying material from the GC. Furthermore, we do not address any other type of attack that may be carried out against the physical link or routing layers. We do not consider active attacks such as physical node capture and compromise.

C. Basic Problems for Key Distribution in Wireless Networks

In this section we present four suitable metrics for the KDP in wireless ad-hoc networks. For each metric, we formulate an optimization problem and present the optimal solution. We show that the formulations for the member key storage and GC transmission metrics reduce to equivalent formulations to wired networks and hence, the same solutions apply. On the other hand, the formulations for the MG transmissions and energy update cost are specific to wireless networks.

1) Member Key Storage, $k(M_i, D)$: Let D denote a key assignment structure to the members of MG. We want to find the optimal key assignment structure D^* that minimizes the average number of keys assigned to each member M_i:

$$D^* = \arg\min_D \frac{1}{N} \sum_{i=1}^{N} k(M_i, D). \quad (1)$$

Note that in (1), the quantity minimized is the average number of keys since key assignment structures need not assign the same number of keys to every member.

Proposition 1: The optimal key assignment structure D^* that minimizes member key storage can be represented as an N-ary key tree, where the GC shares a unique KEK with each member, and the SEK with all members of MG [2], [23], [25].

Proof: Each member needs to hold the SEK in order to decrypt the multicast data. In addition, the GC needs to be able to securely update the SEK to every member in case of a membership change. Hence, each member needs to share at least one pairwise KEK with the GC, to decrypt the SEK update. Thus, the optimal member key storage solution assigns two keys to each member of MG, and can be represented as an N-ary key tree. ■

Note that the optimal solution for the member key storage metric is independent of the nature of the network, wireless or wired. Hence, the solution for wireless networks is the same as the one provided for wired networks in [2], [23], [25].

2) GC Transmissions, $t(M_i, D)$: Let $t(M_i, D)$ denote the number of messages transmitted by the GC when M_i leaves MG, and keys are assigned according to the key assignment structure D. We want to find the optimal D^* that minimizes the average number of key messages transmitted by the GC, to the members of MG, after M_i leaves the group.

$$D^* = \arg\min_D \frac{1}{N} \sum_{i=1}^{N} t(M_i, D) \tag{2}$$

Note that we minimize the average number of GC transmissions required to rekey MG, to take into account unbalanced key assignment structures as well.

Proposition 2: The optimal key assignment structure D^* for member deletions, can be obtained by distributing one KEK to every possible subset of MG [2], [23], [25].

Proof: If each possible subset of MG shares a unique KEK, an arbitrary set of members can be represented by the index of the corresponding KEK. Hence, after the deletion of any set of members, the GC can notify all remaining valid members of MG to use their unique common KEK as the new SEK, by just broadcasting the index of the KEK corresponding to the remaining members. Hence, by assigning a unique key to every possible subset of members, the GC can update the SEK after the deletion of any set of members, by transmitting a single message. ■

As in the case of member key storage, the number of GC transmissions depends on D and not on the network topology. Hence, the optimal solution for wireless networks is identical to the one for wired networks [2], [23], [25].

3) MG Key Update Messages, $m_{M_i}(D)$: Let $m_{M_i}(D)$ denote the number of messages transmitted/relayed by the nodes of the network in order to update the SEK and KEKs after deletion of M_i. We want to find the optimal key assignment structure D^* that minimizes the average number of messages m_{Ave} transmitted/relayed by all network nodes for updating the SEK and KEKs, when a member leaves the group.

$$D^* = \arg\min_D \frac{1}{N} \sum_{i=1}^{N} m_{M_i}(D) \tag{3}$$

In contrast to the previous two metrics, m_{M_i} depends both on the network topology as well as the choice of D. The number of messages the nodes of the network have to transmit/relay after the deletion of a member, varies depending on the specific member being deleted. Thus, we use the average number of MG update messages m_{Ave}, to evaluate the efficiency of a key assignment structure D.

Proposition 3: Finding the optimal key assignment structure D^*, that minimizes the average number of MG update messages m_{Ave}, is an NP-complete problem.

Proof: Under Proposition 2, the GC can update the SEK after the deletion of any set of members from MG, by transmitting a single message to the remaining valid members of MG, when using the optimal structure D^*. Hence, the problem of minimizing the number of messages transmitted/relayed by the network nodes *reduces* to the problem of minimizing the number of messages transmitted/relayed by the nodes of the network to deliver one message from the GC to every member of MG. In turn, the latter problem can be mapped to the problem of finding the minimum power broadcast routing tree R_m rooted at the GC, in which each node of the network can either broadcast a message with unit power $p = 1$, or not transmit at all ($p = 0$). This routing problem is known as the *Single Power Minimum Broadcast Cover problem* (SPMBC) [1], with input parameter $p = 1$ and has been proven NP-complete in [1]. Hence, the problem of minimizing the average number m_{Ave} of MG update messages is also NP-complete. ■

4) Energy Update Cost, $\tilde{E}_{M_i}(D)$: Let $\tilde{E}_{M_i}(D)$ denote the total energy expended by all network nodes, in order to deliver the rekey messages to MG after a member deletion. We want to find the optimal key assignment structure D^*, that minimizes the average update energy E_{Ave}.

$$D^* = \arg\min_D \frac{1}{N} \sum_{i=1}^{N} \tilde{E}_{M_i}(D) \tag{4}$$

The total energy expenditure depends on the network topology and the choice of D. Thus, as was the case for m_{M_i}, $\tilde{E}_{M_i}$ varies depending on which member is deleted from MG. Therefore, we choose the average update energy cost E_{Ave}, to evaluate the performance of D over MG.

Proposition 4: Finding the optimal key assignment structure D^*, that minimizes the average update energy E_{Ave}, is an NP-complete problem.

Proof: Under Proposition 2, the GC can update the SEK after the deletion of any set of members from MG, by transmitting a single message to the remaining valid members of MG, when using the optimal structure D^*. Hence, the problem of minimizing the total energy expenditure required to update the SEK after the deletion of any set of members

reduces to the problem of distributing one message to all valid members of MG, by expending the least amount of energy. The latter problem is equivalent to finding a broadcast routing tree R_E, rooted at the GC, that minimizes the energy required to deliver one message from the GC to every valid member of MG. This problem is known as the *Minimum Broadcast Cover problem* (MBC) [1], [3], a generalized version of the SPMBC problem, for cases where the transmission power level for a node can adopt any value $p \in [0, p_{max}]$. The MBC problem has been proved to be NP-complete in [1], [3] and, hence, the problem of minimizing the average update energy E_{Ave} is also NP-complete. ■

The optimal solution to the member key storage problem, requires the GC to unicast the SEK to each member of MG every time a member joins or leaves MG. Hence, demanding $\mathcal{O}(N)$ number of GC transmissions. On the other hand, the optimal solution to the GC key transmission problem for leave operations requires each user to store at least $2^{(N-1)}$ keys, thus making user storage requirements grow exponentially with group size [2], [23], [25]. Clearly, there is a tradeoff between all four metrics described.

A key assignment structure scalable in both member key storage and GC transmissions was independently proposed in [25] and in [23]. In both proposals it was shown that using a *Logical Key Hierarchy* (LKH) such as a d-ary key tree T, reduces member key storage and GC transmissions to $\mathcal{O}(\log_d N)$. While key trees are minimal structures in terms of member key storage and GC transmissions, not all key trees are energy-efficient [11], [12]. However, we will show that key tree structures designed by incorporating the metrics of m_{Ave} and E_{Ave}, lead to energy and bandwidth-efficient solutions to the KDP for the wireless ad-hoc networks. In the rest of the article, we focus on finding resource-efficient key assignments D that follow the LKH structure T. We now introduce the LKH structure.

D. Logical Key Hierarchies and Key Distribution Trees

We first provide some necessary definitions:

Definition 1 (Node Depth, $r(i)$): The depth $r(i)$ of node i is the length, measured in edges, of the path traced from the node to the root of the tree.

Definition 2 (Node Weight, $w(i)$): The node weight $w(i)$ of node i, is equal to the number of edges leaving i.

Definition 3 (Leaf Ancestor Weight, $w_a(i)$): The leaf ancestor weight $w_a(i)$ of node i is the sum of the weights of all nodes traced on the path from i to the root of the tree.

Figure 1 shows a binary key distribution tree for a network of $N = 8$ nodes, plus the GC. Each node of the tree is assigned a KEK, $K_{l,j}$, where l denotes the tree level, and j denotes the node index. (i.e. $K_{1,2}$ is assigned to node 2 at level 1 of the tree). The root node is at level 0, and K_0 can also be used as the SEK.

In [23], [25], each user is *randomly* assigned to a tree leaf, and holds the keys traced on the path from the leaf to the root of the tree. (i.e. user M_5 in Figure 1 is assigned the set of keys $\{K_{3,5}, K_{2,3}, K_{1,2}, K_0\}$). We denote the subset of users that receive key $K_{l,j}$, as $S_{l,j}$. For example, $S_{1,1} = \{M_1, M_2, M_3, M_4\}$.

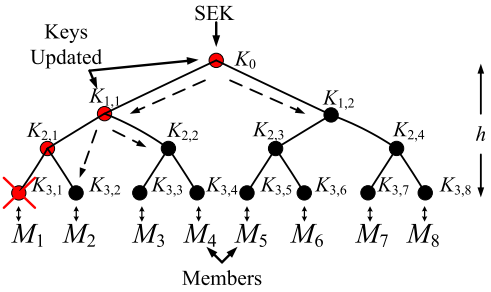


Fig. 1. A binary key tree. Members are placed at the leaf nodes. Each member holds the keys traced on the path from its leaf to the root. As an example M_1 holds keys $(K_0, K_{1,1}, K_{2,1}, K_{3,1})$. If M_1 leaves MG, the GC updates $(K_0, K_{1,1})$ by sending the messages indicated by the dashed arrows.

Under this regime, the number of KEKs stored by each member is equal to the depth of its leaf. Thus, worst-case storage requirements for any node will be $\lceil \log_d N \rceil$ KEKs, and the SEK. Figure 1 shows the keys to be updated if M_1 leaves MG. In this case, the GC will transmit the sequence:

$$\begin{aligned} GC &\rightarrow M_2 : \{K'_{1,1}\}_{K_{3,2}}, \\ GC &\rightarrow \{M_3, M_4\} : \{K'_{1,1}\}_{K_{2,2}}, \\ GC &\rightarrow \{M_2 \sim M_4\} : \{K'_0\}_{K'_{1,1}}, \\ GC &\rightarrow \{M_5 \sim M_8\} : \{K'_0\}_{K_{1,2}}. \end{aligned} \quad (5)$$

Each message in (5) is represented in Figure 1 by a dashed arrow. The arrows leaving $K_{1,1}$ represent the first two messages in (5), while the arrows leaving K_0 represent the last two messages in (5). This illustrates how the number of GC transmissions is proportional to the sum of edges leaving the nodes that correspond to the keys that are being updated. That is, the number of GC transmissions is proportional to $w_a(M_1)$. In fact, it has been shown that GC transmissions due to member deletions increase as a function of $d \log_d N$ [8], [16], [20]. The cost of join operations on the other hand, is proportional to the depth of the leaf the new user is assigned, a function of $\log_d N$ [8], [16], [20].

E. On the Security Properties of LKH Structures

To preserve forward secrecy in LKH structures, the keys common to a deleted member and the remaining valid members of the MG are updated [23], [25]. Furthermore, to preserve backward secrecy, the SEK and relevant KEKs are updated in the event of a new member join [23], [25].

With respect to member collusion, LKH prevents the construction of the set of keys held by any valid member, if the cryptographic keys assigned to members are not correlated, and the GC has a secure channel to individually reach every member of MG [18]. This last property has been referred to in the literature, as the Cover Free Property and has been shown to prevent the collusion of up to any $(N-1)$ members [18]. In LKH, it is sufficient to establish unique pairwise keys between the GC and each member of the MG, to satisfy the cover free property. Interested reader is referred to [18], [23], [25] for a thorough evaluation of the security of LKH.

F. Minimizing Average Update Energy In Key-Tree Structures

While the key-tree assignment structure provides scalability in terms of member key storage and GC transmissions, it has been shown to be energy inefficient when the network topology is not taken into account [11], [12]. In fact, it was shown in [14] that updating the SEK via unicasting to each member (N-ary key tree) can require less energy to rekey a member deletion, compared to a binary tree, despite the larger number of GC transmissions in the case of the N-ary tree. Hence, bandwidth efficiency of the GC may not imply energy efficiency of the network.

Furthermore, it was shown in [12], that an energy-optimal solution for the average update energy does not imply an optimal solution to the number of rekeying messages. Since energy is, in most cases, a more scarce resource than bandwidth, we focus on finding an energy-efficient solution and show that our scheme is also bandwidth-efficient.

In order to rekey the remaining users after a member deletion, the GC has to send rekey messages to multiple subgroups of MG, as shown in (5). Hence, the update energy cost $\tilde{E}_{M_i}$, for rekeying MG after deletion of M_i, is dependent upon the energy required for sending messages to those subgroups. In fact, we can express the update energy cost after the deletion of M_i, as a function of the routing tree R, and the key tree structure T. For simplicity, we have omitted (R,T) from the expressions $E_S(R,T)$:

$$\begin{aligned}\tilde{E}_{M_i}(R,T) &= \sum_{j=1,j\neq i}^{d}(E_{S_{h,j}} + ... + E_{S_{1,j}}) + \sum_{l=1}^{h-1} E_{S_{l,i}\setminus M_i} \\ &= \sum_{l=1}^{h}\sum_{j=1,j\neq i}^{d} E_{S_{l,j}} + \sum_{l=1}^{h-1} E_{S_{l,i}\setminus M_i}, \end{aligned} \tag{6}$$

where $S_{l,j}$ denotes the group of users that share the j^{th} key at the l^{th} tree level, $K_{l,j}$, and h denotes the tree height [11].

Since $\tilde{E}_{M_i}(R,T)$ is a member-specific cost metric, it depends on the member that is being deleted. Therefore, we use the *Average Update Energy*, $E_{Ave}(R,T)$ [11], to evaluate the energy-efficiency of a key-tree distribution structure:

$$\begin{aligned}E_{Ave}(R,T) =& \frac{1}{N}\sum_{i=1}^{N}\tilde{E}_{M_i} \\ =& \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{l=1}^{h}\sum_{j=1,j\neq i}^{d} E_{S_{l,j}} + \sum_{l=1}^{h-1} E_{S_{l,i}\setminus M_i}\right] \\ =& \frac{1}{N}(d-1)\left[\sum_{l=1}^{h} d^{(h-l)}\sum_{j=1}^{d^l} E_{S_{l,j}}\right] \\ &+ \sum_{l=1}^{h-1}\sum_{j=1}^{d^l}\sum_{k=1}^{|S_{l,j}|} E_{S_{l,j}\setminus S_{l,j}(k)}. \end{aligned} \tag{7}$$

Since the routing tree R is determined by more costly network functions such as routing, we want to minimize $E_{Ave}(R,T)$, expressed in (7), with respect to T:

$$T^* = \arg\min_{T} E_{Ave}(R,T). \tag{8}$$

In the following section we analyze the difficulty of finding the optimal tree T^*.

III. On the Difficulty of Finding an Optimal Key Distribution Tree

Finding a solution to (8) implies finding an optimal allocation of members to leaf nodes, so that the sum of energy costs to multicast to subgroups that share common keys is minimized. To our knowledge, there is no algorithm that yields an optimal key tree, or a characterization of the complexity of the problem in (8).

While building a tree using hierarchical clustering is one approach, the authors of [11] show a divide and conquer strategy for constructing each level of the key tree. Finding a minimum energy partition of MG into subgroups of size d^i, and using that partition to construct a minimum energy partition of MG into subgroups of size d^{i+1}, will not lead to an optimal solution for (8).

It is possible, however, to use this divide and conquer approach to build a sub-optimal solution. In [11], the authors show that finding the optimal partition of MG into subgroups of $d = 2$ users, in order to construct just one level of the key tree T, is equivalent to solving the *Minimum Weight Non-Bipartite Matching Problem* (MWNBM) [4].

Similarly, finding an optimal partition of MG into subgroups of $d = k$ ($k \geq 2$) members, is equivalent to solving the *Minimum Weight Non k-partite Matching* problem (MWNkPM). While the MWNBM has a polynomial solution [4], [7], [10], we now show that the MWNkPM is NP-hard for all $k \geq 3$:

Proposition 5: The Minimum Weight Non k-partite Matching problem is NP-hard for $k \geq 3$.

Proof: The proof is provided in Appendix I. ■

In the following section, we present VP3, a heuristic that employs cross-layer information and a divide and conquer approach, to build an energy and bandwidth efficient solution for the KDP in wireless ad-hoc networks.

IV. VP3: Vertex-Path, Power-Proximity, A Cross-Layer Approach

The VP3 algorithm borrows its name from the network and physical layer information it exploits, in order to build an energy-efficient key distribution tree; **V**ertex-**P**ath, **P**ower-**P**roximity (VP3). The concept of *Power-proximity* was first introduced in [11], and will be used throughout this paper:

Definition 4 (Power-proximity): Node j^* is said to be in power-proximity to node i over the set S, if $j^* = \arg\min_{j\in S}|P_i - P_j|$.

We first introduce the main ideas of VP3, and then present algorithmic details. VP3 reduces E_{Ave} by constructing key trees that assign the same KEKs to members that receive messages via common routing paths. For instance, if a member M_i lies on the path from the GC to member M_j, and a message is sent to both M_j and M_i, the latter will receive

the message for free. Hence, by assigning a common KEK, $K_{k,l}$ to subgroup $S_{k,l} = (M_i, M_j)$, VP3 decreases the energy expenditure required for updating the SEK and common KEKs, whenever transmitting a message to both nodes.

To explore this idea, VP3 discovers which members of MG share the longest paths or, equivalently, which members have paths that differ the least, a property that is extracted from a given broadcast routing tree R. The network paths from the GC to each node are represented as binary codewords of length equal to N. The k^{th} position of the i^{th} codeword $C_i(k)$, has a value of one if node k has to transmit in order for a message unicasted by the GC to reach node i, and a zero otherwise[1]. Thus, the length of a path from the GC to node i, PA_i, can be obtained by computing the *Hamming Weight* $H_w(C_i)$ of the codeword C_i that represents PA_i [22]:

$$H_w(C_i) = \sum_{k=1}^{N} C_i(k). \tag{9}$$

Once codewords have been constructed for each node, we need a metric that allows us to measure the *path distance*, defined below, between the paths of any two nodes:

Definition 5 (Path Distance): We define the path distance between two nodes i, j as the difference between the unicast paths from the GC to i, j, in number of nodes. We measure the path distance between i and j, by computing the *Hamming Distance* $H_d(i, j)$, between the codewords corresponding to the unicast paths to i, j [22]:

$$H_d(i, j) = \sum_{k=1}^{N} C_i(k) \oplus C_j(k). \tag{10}$$

A. The VP3 Algorithm

We assume two sets of parameters as inputs: (a) the NxN binary connectivity matrix C, where each row C_i is a codeword that represents the node path from the GC to node i, such that entry $C_i(k) = 1$ if node $k \in PA_i$, and $C_i(k) = 0$ otherwise and, (b) a vector E of length N, where the i^{th} entry E_i, indicates the energy expenditure required to unicast a message from the GC to node i, following the path indicated by the connectivity matrix C. To construct a d-ary key distribution tree, we execute the following steps:

Step 1: Calculate the Hamming weight $H_w(C_i)$ for each row in C, corresponding to the path from the GC to node i.

Step 2: Choose the node i^* with the *maximum Hamming weight* $i^* = \arg\max_{i \in MG}(H_w(C_i))$. If there is more than one node that satisfies this condition then, from this list, pick the node i^* to be the one with maximum E_i.

Step 3: Pick the $(d-1)$ nodes with the shortest Hamming distances $H_d(i^*, j), j \in MG \backslash i^*$. If there are more than $(d-1)$ nodes with equal $H_d(i^*, j)$ always pick first, if any, the node or nodes found on the path from the GC to i^*. For the remaining nodes, pick those with the largest E_j. Assign a unique KEK to all members chosen in this step.

[1]Construction of the codewords is equivalent to generating the connectivity matrix for the network.

Step 4: Repeat Steps 2, 3 until all nodes belong in subgroups of at most d nodes and are assigned a unique KEK.

Step 5: Generate a matrix C' with rows corresponding to the subgroups generated in Step 4 and columns corresponding to the network nodes. An entry $C'_i(k) = 1$ if node k is traversed by the path from the GC to any of the members of subgroup S_i, and $C'_i(k) = 0$ otherwise. Compute the vector E', the i^{th} entry of which indicates the energy expenditure required to multicast a message from GC to all members of S_i, following the paths indicated by the connectivity matrix C'. Execute Steps $1 \sim 4$ with inputs C', E'.

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

B. Applying VP3 on a Sample Network

We now present the application of VP3 on the sample network of Figure 2(a). The numbers on the links indicate the energy link cost. Nodes $1 - 8$ correspond to members $M_1 - M_8$ of MG. Figure 2(c) shows the connectivity matrix C for MG, the Hamming weights $H_w(C_i)$ for each row C_i, and the energy expenditure E_i necessary to send a message from the GC to member M_i.

We want to construct a binary key tree ($d = 2$) using VP3. Column H_w in Figure 2(c) shows the result of executing Step 1. Step 2, identifies node $i^* = 5$ as the node with the greatest H_w, and withdraws it from the pool.

Using Step 3, VP3 finds nodes $\{7, 2\}$ to have the shortest H_d to 5. Since we need to choose only one node $(d = 2)$, and 7 is on the path from GC to 5, $\{M_5, M_7\}$ are assigned a unique KEK, and node 7 is removed from the pool. Note that because node 7 lies on the path from the GC to node 5, the choice made by VP3 maximizes the length of the common path over the set of available choices, nodes 2 and 7.

In Step 4, VP3 repeats Steps 2, 3; nodes $\{2, 6, 8\}$ have the highest H_w, and 6 is selected since it has the highest E_i. Since node 8 has the smallest H_d to 6, nodes 6, 8 are paired and $\{M_8, M_6\}$ are assigned a unique KEK. Similarly, VP3 groups $\{M_2, M_3\}$ and $\{M_1, M_4\}$ and a unique KEK is assigned to each group.

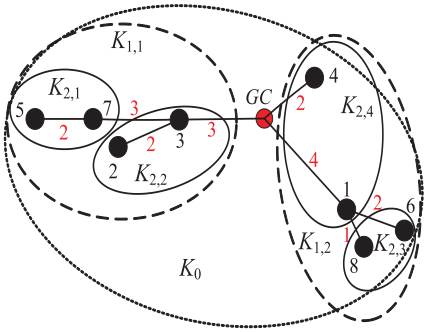
In Step 5, VP3 recomputes the connectivity matrix C' and energy matrix E' for the pairs generated in Step 4, and repeats Steps 1 to 4. Nodes $\{2, 3, 5, 7\}$, $\{1, 4, 6, 8\}$ are grouped, and members $\{M_2, M_3, M_5, M_7\}$, $\{M_1, M_4, M_6, M_8\}$, are assigned unique KEKs, respectively. At this point the SEK is assigned to all members and the key tree construction is completed. Figure 2(b) presents the key distribution tree.

C. Balancing Trees for Improved Energy-Efficiency

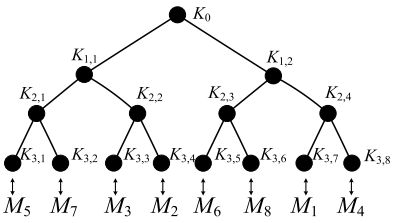
In [16], Moyer et al. define the concept of *balanced trees* and show that maintaining such trees ensures that GC transmissions during rekey operations are kept at $\mathcal{O}(d \log_d N)$.

Definition 6 (Balanced Tree): A tree is said to be balanced, if leaf depth differs by at most one between any two leaves of the tree [16].

We note that, depending on the size of MG, the application of VP3 may yield an unbalanced tree . Unbalanced trees have



(a)

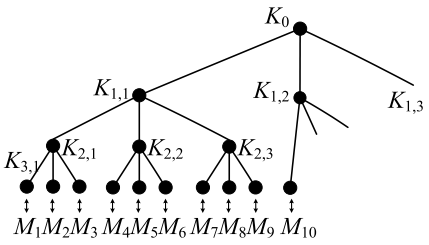


(b)

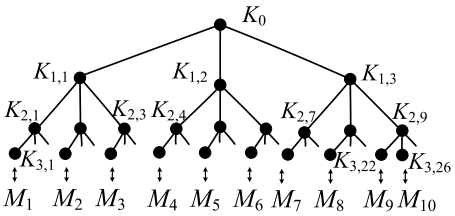
	1	2	3	4	5	6	7	8	H_w	E
1	0	0	0	0	0	0	0	0	0	4
2	0	0	1	0	0	0	0	0	1	5
3	0	0	0	0	0	0	0	0	0	3
4	0	0	0	0	0	0	0	0	0	2
5	0	0	1	0	0	0	1	0	2	8
6	1	0	0	0	0	0	0	0	1	6
7	0	0	1	0	0	0	0	0	1	6
8	1	0	0	0	0	0	0	0	1	5

(c)

Fig. 2. (a) The broadcast routing tree for an ad-hoc network of eight nodes plus the GC. Nodes $\{1-8\}$ are members of MG. The numbers on the links indicate the units of energy required to transmit a message through that link. The ovals indicate the grouping of the members into the key tree after the execution of VP3. (b) The key distribution tree constructed by VP3 for the network in Figure 2(a). (c) The Connectivity Matrix for the network in Figure 2(a). The first row and first column denote the node ID, column 10 denotes the Hamming weight of each codeword, and the last column denotes the energy required to unicast a message to each node.



(a)



(b)

Fig. 3. (a) An unbalanced ternary key tree of $N = 10$, $\overline{w}_a(T) = 7.5$. (b) Balancing the tree reduces $\overline{w}_a(T)$ to 7.2.

been shown to require more GC key transmissions, since their average leaf ancestor weight $\overline{w}_a(T)$, is larger compared to that of balanced trees [8], [16], [20]. Hence, unbalanced trees, on average, require more energy for rekeying, as will be shown in Section VI.

We now illustrate how the use of balanced trees reduces $\overline{w}_a(T)$ in a tree, which leads to savings in GC transmissions. Figure 3(a) shows an unbalanced ternary key tree for a ten node network, in which the empty branches at levels 0 and 1 are left indicated. The ancestor weight $w_a(M_i)$, for the leftmost nine leaves is eight, $w_a(M_{10}) = 3$, and $\overline{w}_a(T) = 7.5$. By contrast, Figure 3(b) shows a balanced tree with $w_a(M_i) = 7, i \in \{1, ..., 8\}$, $w_a(M_9) = w_a(M_{10}) = 8$, and $\overline{w}_a(T) = 7.2$. Nevertheless, an analysis of Figures 3(a) and 3(b) reveals that the subgroups in both representations are mostly unaffected. For example, subgroups $S_{2,1}$ and $S_{2,2}$ in Figure 3(a), have the same members as $S_{1,1}$ and $S_{1,2}$ in Figure 3(b).

Our simulation results show that balancing trees has significant impact on energy consumption as N increases. Hence, we have modified VP3 to always construct balanced trees without affecting the efficiency of the resulting partitions of MG into subgroups of the desired size. We do this by distributing members among the branches of T, as evenly as N will allow. If an even distribution of members among the branches of T is not feasible, we favor grouping of the remaining members into the subgroups with shortest paths. We now describe the algorithmic steps involved in balancing the tree.

Before building the key tree structure, we calculate the number of members that should be assigned to each subgroup at level h of the tree. This is done by computing the number of branches B, in the balanced tree at level $(h-1)$, $B = d^{\lceil \log_d N \rceil - 1}$. We then assign $g = \lfloor \frac{N}{B} \rfloor$ members to each subgroup at level h. The remaining $L = N - gB$ members are assigned one to each of the last L subgroups to be formed by the first iteration of VP3. For example, for the tree in Figure 3(b), the number of subgroups at level h is equal to the number of nodes in the balanced tree at level $(h-1)$, $B = 3^{(\lceil log_3 10 \rceil - 1)} = 9$, and each subgroup will have at least $g = \lfloor \frac{10}{9} \rfloor = 1$ node. We then have $L = 10 - (1)(9) = 1$ node left (M_{10}), which is assigned to the last group formed by the first iteration of the algorithm, $S_{2,9}$. Note that the added computational cost of balancing the trees is that of computing three quantities: B, g and L.

In Figure 4, we present the pseudo-code for VP3, including the balanced tree modification. The $ConnectivityMatrix()$ function computes the connectivity matrix for its argument set. The $EnergyMatrix()$ function 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 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 KEK to every element of the argument set.

D. Algorithmic Complexity of VP3

The algorithmic complexity of VP3 is determined by the complexity of its subgrouping process. The algorithm first identifies the codeword C_{i^*}, with the largest Hamming weight, then computes the Hamming distances from all other codewords to C_{i^*}, and picks the $(d-1)$ codewords with the shortest

$C = ConnectivityMatrix(MG)$, $E = EnergyMatrix(MG)$
$B = d^{(\lceil \log_d N \rceil - 1)}$, $g = \lfloor \frac{N}{B} \rfloor$
for $l = 1 : \lceil \log_d(N) \rceil$
  $H_w(i) = \sum_{j=1; j \neq i}^{N} C_i(j), \forall$ rows C_i
  for $k = 1 : B$
    $i^* = \arg\max_{i \in MG} H_w(i)$
    if $|i^*| > 1$ **then** $i^* = \arg\max_{i \in i^*} E_i$
    $MG = MG \backslash \{i^*\}$
    $j' = \{j \in MG \ni \arg\min_{j \in MG} H_d(i^*, j)\}$
    if $l > 1$ **then** $gs = d$, **else** $gs = g$
    if $l = 1$ **and** $k \geq N - gB$ **then** $gs = gs + 1$
    if $|j'| > (gs - 1)$ **choose** j' path $GC \rightarrow i^*$
      and $(gs - 2) \in j' \ni \arg\max_{i \in j'} E_i$
    $MG = MG \backslash \{j'\}$, $G = G \cup j'$
    $AssignKey(j')$
  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.

Hamming distance to C_{i^*}. This process has to be repeated $\lceil \frac{N}{d^j} \rceil$ times at each of $(h-1)$ tree levels, where $j = h - i$, and i is the tree level being built. The total number of operations for this process is:

$$\sum_{j=0}^{h-1} \sum_{i=0}^{\lceil \frac{N}{d^j} \rceil - d} \left[(d+2) \left(\left\lceil \frac{N}{d^j} \right\rceil + id \right) - d - 1 \right] < d^3 N^2. \quad (11)$$

Since in general, we are interested in trees with small d, the worst case algorithmic complexity of VP3 is $\mathcal{O}(N^2)$.

E. An Analytical Bound for Subgroup Choices in VP3

In this section we evaluate the deviation of a subgrouping choice made by VP3 from the optimal choice, by computing the worst case *cumulative path divergence* $\Delta(S)$, defined below, for a subgroup S, of arbitrary size, with subgroup head $\alpha(S)$:

Definition 7 ((Sub)group Head, $\alpha(S)$): We define the (sub)group head $\alpha(S)$, of a (sub)group S, as the (sub)group member that satisfies $\alpha(S) = \arg\max_{i \in S}\{E_i\}$.

Definition 8 (Cumulative Path Divergence): The cumulative path divergence $\Delta(S)$ of a subgroup S with subgroup head $\alpha(S)$, is defined as:

$$\Delta(S) = \sum_{k=1}^{N} \left[\overline{C}_{\alpha(S)} \bigwedge \left(\bigvee_{j \in S, j \neq \alpha(S)} C_j \right) \right], \quad (12)$$

where the symbol $\vee$ denotes successive bitwise OR operations over the codewords of all members of the set $S \backslash \alpha(S)$, the symbol $\bigwedge$ denotes a single bitwise AND operation, and $\overline{C}_i$ denotes the complement operation on codeword C_i.

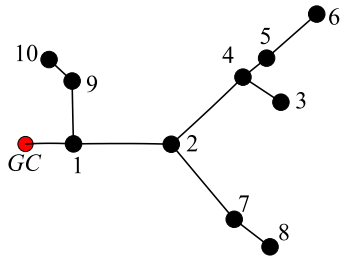


Fig. 5. The cumulative path divergence between nodes 6 and 8 is $\Delta(6,8) = 1$. Note that the common path between nodes 6 and 8 goes from the GC to node 2 and $\Delta(2,6) = 0$.

$\Delta(S)$, expresses the number of additional transmissions required by the network, so that a multicast message sent by the GC reaches all members of $S \backslash \alpha(S)$, once the subgroup head $\alpha(S)$ has already been reached.

As an example, in Figure 5 the path distance between nodes $\{2,6\}$ is $H_d(2,6) = 3$, but their path divergence is $\Delta(2,6) = 0$, since node 2 is in the path from GC to node 6, and is reached for free whenever a message is sent to node 6. Similarly, for $S = \{6,8,10\}$, $\alpha(S) = 6$, and $\Delta(S)$ can be computed as:

$$C_6 = 1101100000, \overline{C}_6 = 0010011111,$$
$$C_8 = 1100001000, C_{10} = 1000000010,$$
$$\Delta(S) = \sum_{k=1}^{10} \left[\overline{C}_6 \wedge (C_8 \vee C_{10}) \right] = 2.$$

$\Delta(S) = 2$ denotes the two additional transmissions $7 \rightarrow 8$ and $9 \rightarrow 10$ required to deliver a message m to nodes 8 and 10, when m is sent to 6.

VP3 aims to reduce the energy cost of the key distribution tree by maximizing energy savings when building a partition of MG into subgroups of size d. The idea is to maintain total subgroup cost E_S, as close to the unicast cost of the subgroup head as possible. Thus, we consider a subgroup S achieves optimal cost if $E_S = E_{\alpha(S)}$, where $\alpha(S) = \arg\max_{i \in S}\{E_i\}$.

It is important to point out that $E_S = E_{\alpha(S)}$ implies that $\Delta_S = 0$, i.e., no additional transmissions are required to reach any of the members in $S \backslash \alpha(S)$, when $\alpha(S)$ is reached. Hence, $E_S = E_{\alpha(S)}$ indicates that a message multicasted from the GC to S will be relayed with the *minimum number of MG update messages* for the given routing tree R. That is, $E_S = E_{\alpha(S)}$ implies optimal energy update cost *and* optimal MG update messages when transmitting a message to S, for fixed R.

We note that while $E_S = E_{\alpha(S)}$ implies $\Delta(S) = 0$, the converse is not true, as can be shown by the example of Figure 5. Let $S = \{3,5,6\}$, and $E_{4 \rightarrow 3} > E_{4 \rightarrow 5}$. In that case, though $\Delta(S) = 0$, we $E_S = E_6 - E_{4 \rightarrow 5} + E_{4 \rightarrow 3} > E_6$.

We now calculate the worst case $Delta_d(S)$ for a subgroup S of size d, when S is generated using VP3:

Proposition 6: The maximum cumulative path divergence $\Delta_d^*(S)$, for a subgroup S of size $d > 2$ is:

$$\Delta_d^*(S) = (d-1) \max H_w(i), i \in R.$$

Proof: Can be found in Appendix II. ■

$J = JoiningGroup$, $D = LeavingGroup$
$T' = UnassignedLeavesinT$
$C = ConnectivityMatrix(MG)$, $E = EnergyMatrix(MG)$
$T = T \backslash D$
for $k = 1 : |J|$
  $\alpha(J) = \arg\max_{i \in J} H_w(i)$
  if $|\alpha(J)| > 1$ **then** $\alpha(J) = \arg\max_{i \in \alpha(J)} E_i$
  $nn(\alpha(J)) = \{k \in MG \backslash J \ \ni \ \arg\min_{k \in MG \backslash J} H_d(\alpha(J), k)\}$
  if $|nn(\alpha(J))| > 1$ **then** $nn(\alpha(J)) = \arg\max_{k \in nn(\alpha(J))} E_k$
  $t' = \{t \in T' \ \ni \ \arg\min_{t \in T'} \lambda(nn(\alpha(J)), t)\}$
  $AssignLeaf(t', \alpha(J))$
  $J = J \backslash \alpha(J)$
endfor

Fig. 6. Pseudo-code for On-line 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 $AssignLeaf()$ function assigns leaf t' to node i^*.

V. On-line VP3

We expect wireless ad-hoc networks to be dynamic environments in which different members join and leave MG at different times. Thus, we need a method that allows addition and deletion of members, without having to reconstruct T every time there is a change in group membership. This is important since our simulations show that tree reconstruction costs are at least an order of magnitude higher than updating the key tree, whenever $N \geq 100$.

In Section IV, we proposed the use of balanced trees to reduce E_{Ave}. Balancing the trees may leave a number of unassigned leaves, depending on group size N. On-line VP3 takes advantage of these empty spaces, and tries to find the best available leaves to insert new members.

Two different update strategies have been proposed for dynamic maintenance of key trees: individual rekeying [25], and batch rekeying [15], [19], [26], [27]. Individual rekeying or immediate rekeying of MG after each join and leave, generates a significant amount of GC transmissions when group membership changes rapidly. Because of this, we adopt a batch rekeying strategy for On-line VP3, where the GC has a rekeying interval during which all join and leave requests are collected, and executed at the end of the interval.

In On-line VP3, once the rekeying interval expires, nodes leaving MG simply vacate their leaves, which are then made available to incoming members. New members joining MG are first inserted into the routing tree R. The GC uses the updated R to identify the best empty leaf in the tree T, and assigns that leaf to the joining member. This decision is based on the concepts of *nearest neighbor* and *leaf distance*.

Definition 9 (Nearest Neighbor, $nn(i)$): Node j is the nearest neighbor of node i, in R, if: (a) j lies on the path from GC to i, or i, j have a common parent node, and (b) i and j are in power-proximity, over all nodes that satisfy (a).

Definition 10 (Distance Between Two Leaves in T, $\lambda(i, j)$): The distance $\lambda(i, j)$ between two leaves i and j in T, is the length of the shortest path from i to j, over T.

A. Description of On-line VP3

The initial input for On-line VP3 is: (a) the key tree created by VP3, (b) the list J of joining members, (c) the list D of members that are being deleted from MG, (d) the NxN connectivity matrix C, for the new multicast topology and, (e) the vector E of length N, denoting the energy expenditure required to transmit a message from GC to each node i. On-line VP3 executes the following steps:

Step 1: Delete all members of set D from the key tree.

Step 2: Select the group head of J, $\alpha(J)$, and find its nearest neighbor $nn(\alpha(J))$.

Step 3: Find the unassigned leaf i^* in T, with the shortest distance $\lambda(i^*, k)$, where k is the leaf assigned to $nn(\alpha(J))$. Assign leaf i^* to $\alpha(J)$, and erase $\alpha(J)$ from J.

Step 4: Repeat steps 2 and 3 until all new members have been assigned a leaf in the key tree.

For example, suppose that $J = \{M_{11}, M_{12}\}$, join a multicast group that has the key tree structure shown in Figure 3(b). Assume that after executing steps 1 and 2, On-line VP3 has determined that $\alpha(J) = M_{11}$, and that $nn(M_{11}) = M_8$. Node M_{11} will now be assigned leaf node $K_{3,23}$. Now suppose that $nn(M_{12}) = M_9$, then M_{12} will be assigned leaf node $K_{3,27}$.

We note that Step 3 looks for $nn(\alpha(J))$, among all members of MG that are not in J. This is done to ensure that the $nn(\alpha(J))$ that is picked by On-line VP3 will indeed, have a leaf assigned in T.

In Figure 6, we present the pseudo-code for On-line VP3. As was the case with VP3 in the previous section, the $ConnectivityMatrix()$ function computes the connectivity matrix for its argument set. The $EnergyMatrix()$ function 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. The $AssignLeaf()$ function assigns a tree leaf to a member.

B. Algorithmic complexity of On-line VP3

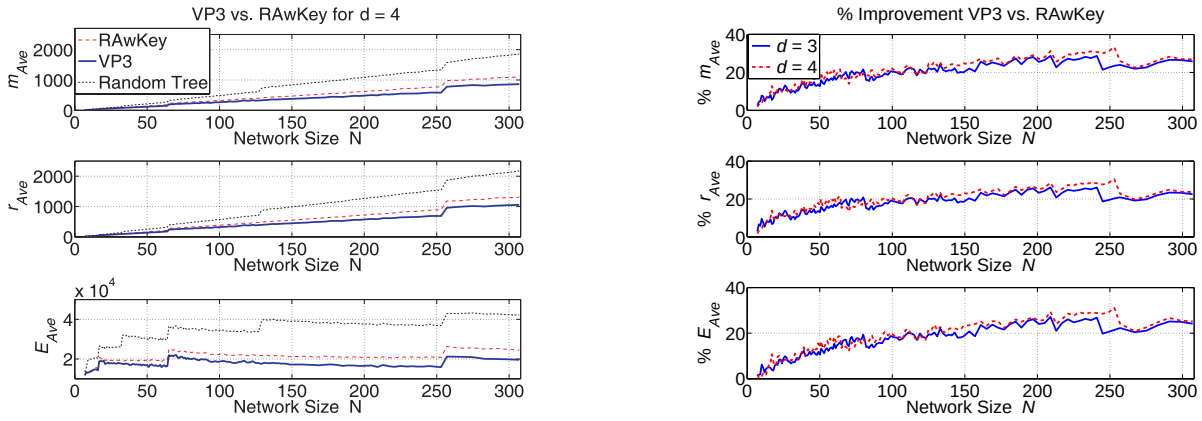
The main tasks executed by On-line VP3 are (a) the computation of the Hamming distance between a joining node's codeword and those of the existing members of MG, (b) finding each joining node's nearest neighbor in R, and (c) finding the corresponding nearest available leaf in T. Each of these operations has complexity $\mathcal{O}(N)$.

Therefore, worst case algorithmic complexity of On-line VP3 is $\mathcal{O}(N)$, which is similar to the complexity of simply looking for the first available leaf in the tree and inserting.

VI. Performance Evaluation

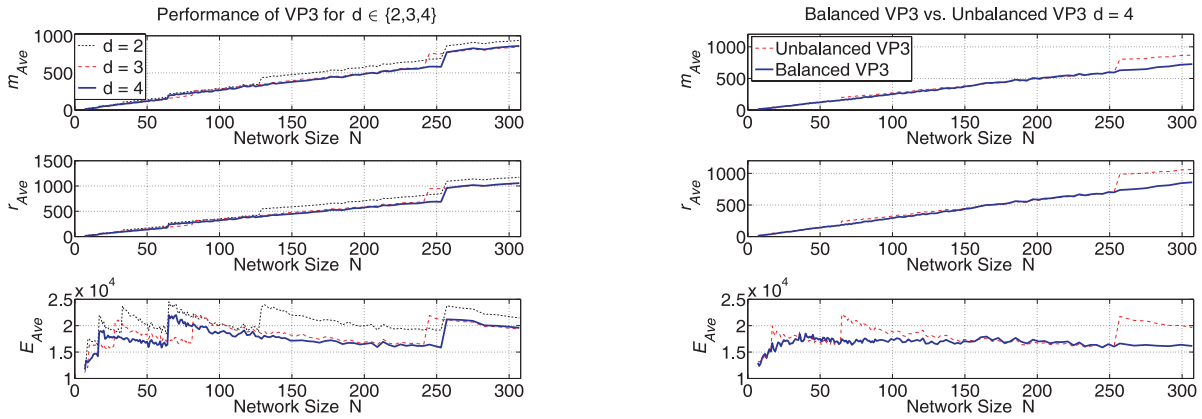
To evaluate the performance of VP3, we generated random network topologies confined to a region of size 100x100. Following the network generation, we used the Broadcast Incremental Power (BIP) algorithm [24] to construct and acquire the routing paths from the GC to every group member [2]. Nodes were assumed able to adjust their transmission power

[2] Any other suitable routing algorithm can be applied as well [1], [24].



(a) (b)

Fig. 7. A comparison between the VP3, RAwKey and the random key tree algorithm. (a) The graph on top shows the average number of MG update messages, the graph in the middle shows the average number of receptions, and the graph at the bottom shows average update energy. Each data point is the average result over 100 randomly generated networks. (b) % of improvement in all three measures m_{Ave}, r_{Ave} and $EAve$ obtained by VP3 over RAwKey for different sizes of MG.



(a) (b)

Fig. 8. (a) Comparison in performance of VP3 for trees of degree $d \in \{2, 3, 4\}$. The graph on the top shows m_{Ave}, the graph in the middle shows r_{Ave} and the graph on the bottom shows E_{Ave}. (b) Average MG update messages, average number of receptions and average update energy for different multicast group sizes, for balanced and unbalanced trees.

in order to route messages to any node dictated by the decision process of BIP [24]. Furthermore, we assumed a collision free environment where no packet losses occur due to multiple access in the common channel. Finally, we assumed that nodes were always available to forward key update packets.

A. Energy model

We assumed that the energy required to send a message to a node at a distance ℓ from a source is proportional to the square of ℓ (free path propagation). For simplicity we set the proportionality constant equal to one. Hence, $E(\ell) = \ell^2$ Energy Units (E.U.) We also assumed that the energy required to receive a message is the sum of the fixed cost of powering the antenna plus the fixed processing cost. For simplicity we set the reception energy expenditure to 1 E.U. This assumption allows us to scale the energy expenditure due to reception, since the receiving energy becomes equal to the number of messages received in the whole network.

B. Comparison between VP3 and RAwKey

In our first experiment, we compared VP3 with the previously known best algorithm for building key tree structures for wireless ad-hoc networks, RAwKey [11]. RAwKey provides a low-complexity solution for incorporating the routing information into the key tree construction [11]. Initially, the energy required to unicast a single message to every member of MG based on the routing tree R, is organized in an ascending order sorted list. Then, the members of the MG are placed in the leaves of the key tree T in the same order as the order in the unicast energy sorted list. Compared to VP3, RAwkey does make explicit use of the routing paths and, hence can result in inefficient member groupings [13].

We also compared VP3 with a random key assignment algorithm as in wired networks [23], [25]. Since for a fixed key tree degree d, the key assignment structures built by VP3, RAwKey, and the random key tree algorithm have the same member storage and GC transmissions requirements, we compared the three methods in terms of average MG update messages m_{Ave}, average number of receptions r_{Ave}, and average update energy E_{Ave}.

Figure 7(a) shows the m_{Ave} (top graph), r_{Ave} (middle graph) and E_{Ave} (bottom graph), for trees of degree $d = 4$ and for different multicast group sizes N. All trees were left

unbalanced. Due to space limitations, we have omitted the results for binary and ternary trees. In the top and middle graphs of Figure 7(a), we observe that sudden increases in m_{Ave} and r_{Ave} occur when $N = d^i+1, i \in \mathbb{Z}^+$. The increases in m_{Ave} and r_{Ave} are a consequence of leaving the key tree unbalanced, since in that case the average leaf ancestor weight $w_a(M_i)$ significantly increases for those nodes with large Hamming weight H_w during the transition from $N = d^i$ to $N = d^i + 1$. As we discussed in Sections II and IV, an increase in $w_a(M_i)$ for those nodes with large H_w implies an increase in the number of GC transmissions directed to nodes with longer paths, which in turn leads to an increased number of relaying messages, which in turn increases the number of receptions.

The bottom graph of Figure 7(a) shows the E_{Ave} for different multicast group sizes N. We observe that the sudden increases in m_{Ave} and r_{Ave} from the top and middle graphs of Figure 7(a), translate into sudden increases in E_{Ave}, also due to the use of unbalanced trees. As N continues to increase, however, $\overline{w}_a(T)$ decreases, and E_{Ave} is reduced. This happens because the size of the deployment area is fixed. Thus, as N increases and the nodes become more densely packed, the number of relaying messages required to rekey MG increases, but the average energy cost per relayed message decreases.

Figure 7(b) shows the performance improvement achieved by VP3, over RAwKey, on all three parameters m_{Ave}, r_{Ave} and E_{Ave}, for key trees of degree $d \in \{3, 4\}$. While average improvement on all metrics is 14%, the average for networks of size $N \geq 150$ increases to 20%. The difference in performance between VP3 and RAwKey occurs due to the near optimal decision process of VP3 when compared to RAwKey, which ignores path direction [11], [13].

Figure 8(a) compares m_{Ave}, r_{Ave} and E_{Ave} for key trees of degree $d \in \{2, 3, 4\}$, generated using VP3. We note that binary trees are clearly outperformed by ternary and quaternary trees, which in turn perform quite similarly for the selected sizes of MG. This happens because the number of GC transmissions increase much more rapidly for binary trees, due to the increase in tree height. Nevertheless, the trend is inverted for $d > 4$, because the increase in subgroup size d implies an increase in the number of unicast transmissions required for rekeying. This increase outweighs reductions due to shorter tree height. Our simulations show that the best results are obtained when we use key trees of degree $d \in \{3, 4\}$.

C. Effect of the Use of Balanced Tree Topologies with VP3

In our second experiment, we evaluated the effect of balancing the key tree structures, as described in Section IV.

The top and middle graphs in Figure 8(b) shows the effect of balanced tree topologies on m_{Ave} and r_{Ave}. We observe that both parameters grow almost linearly with N. This is to be expected, since the MG update messages required to complete rekey operations are not bounded by the size of the area in which networks were generated.

The bottom graph in Figure 8(b) shows the improved E_{Ave} achieved by VP3 when balancing the tree structure. E_{Ave} is almost constant for networks of size $N \geq 50$, both for ternary and quaternary trees. The size of the deployment area is fixed, thus, as N increases and the nodes become more densely packed, the number of MG update messages increases, but the average energy cost per message decreases. Since VP3 provides near optimal grouping of members, the increase in relay messages does not increase E_{Ave}.

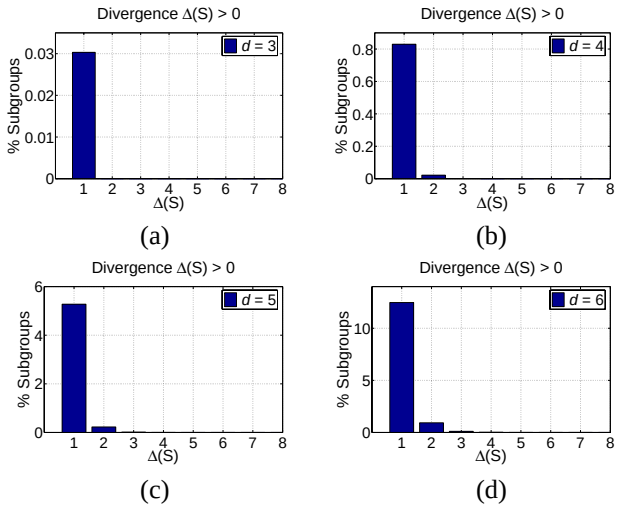


Fig. 9. $\Delta(S)$ observed in 29,300 randomly generated networks. Networks of size $N \in [8, 300]$ were generated at random, 100 networks for each size. The histograms show the percentage of subgroups of size $d \in \{3, 4, 5, 6\}$ that showed $\Delta(S) > 0$, over the *total* number of subgroups that were formed by VP3, for all networks.

D. Path Divergence of VP3

For our third experiment, we generated 100 networks for each network size $N \in [8, 300]$ (a total of 29,300 networks), in an area of 100x100. We then employed VP3 to partition each network into groups of size $d \in \{3, 4, 5, 6\}$ (steps $1-4$ of VP3) and evaluated $\Delta(S_i)$ for each of the resulting subgroups, using (12)[3].

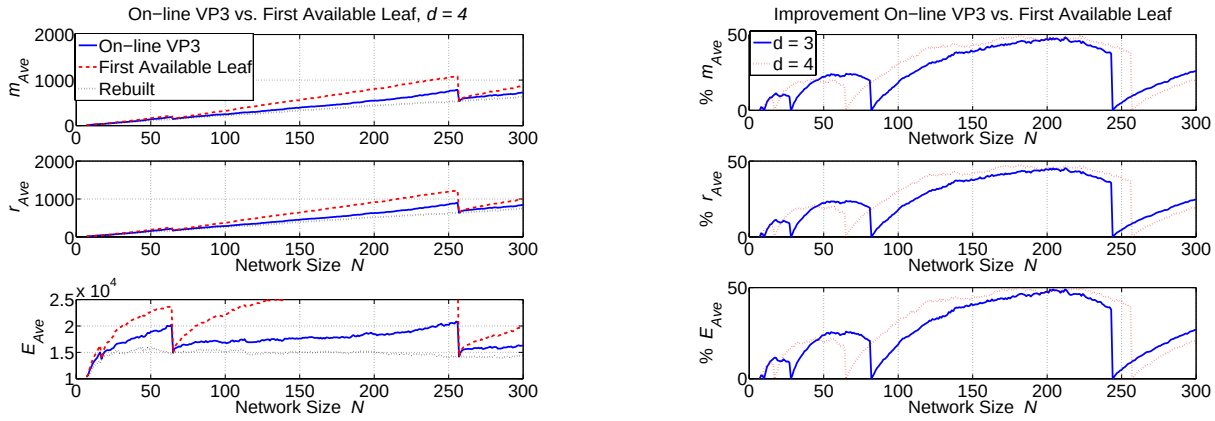
The histograms in Figure 9 present the percentage of subgroups that showed a $\Delta(S_i) > 0$, for subgroups of different size. As an example, in Figure 9(a) only 0.03% subgroups of size $d = 3$ out of the subgroups formed from the 29,300 networks tried, had $\Delta(S) > 0$.

Note that while the worst-case bound indicates that $\Delta_3^*(S) = \max H_w(i)$, the conditions required to achieve this divergence occur in the specific network topology shown in the Figure 13 in Appendix II, and all its isomorphics. In fact, none of the subgroups obtained in our simulations exceeded $\Delta(S) = 1$, for $d = 3$, and we did not find a case in which $\Delta(S) > 7$, for $d \in \{3, 4, 5, 6\}$. Our simulations suggest that the worst-case bound in (12) may be overly pessimistic for most networks, and that the vast majority of groups generated by the decision process of VP3 have zero path divergence.

E. Performance of On-line VP3

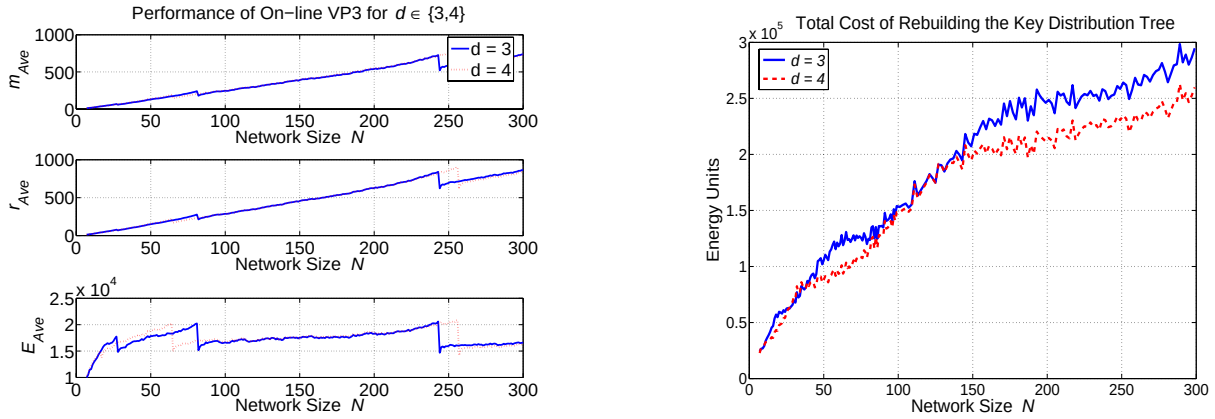
In our fourth experiment, we evaluated the performance of On-line VP3. To do this, we generated small networks of 8

[3]For $d = 2$ it can be proved analytically that VP3 partitions each network into subgroups with $\Delta(S) = 0$.



(a) (b)

Fig. 10. A comparison between the On-line VP3, insertion in the first available leaf and tree reconstruction. The graph on the top shows m_{Ave}, the middle graph is r_{Ave}, and the graph on the bottom shows E_{Ave}. (b) Performance improvement when using On-line VP3 over choosing the First Available Leaf for insertion.



(a) (b)

Fig. 11. (a) Comparison between ternary and quaternary trees, using On-line VP3. (b) Cost of rebuilding the tree after each member join, averaged over 100 network topologies.

nodes in an area of 100x100, and then added nodes one at a time, until $N = 300$. At each step, the joining node was added to the physical network, and the broadcast routing topology R, was recomputed using BIP [24]. Once R was updated, we inserted the new node into the original key distribution tree using two different methods: (a) selection of the best available tree leaf, chosen by On-line VP3 as described in Section V, and (b) searching the key tree from left to right, and inserting the new member in the first available leaf. We then computed m_{Ave}, r_{Ave} and E_{Ave} for the resulting key tree.

Figure 10(a) shows the results for a tree of degree $d = 4$. The upper graph shows m_{Ave}, middle graph is r_{Ave}, and the bottom graph shows E_{Ave}. The abrupt drops in all graphs take place when network size is $N = d^i + 1, i \in \mathbb{Z}^+$ (tree height increases by one), at which point we rebuild the tree in order to always maintain an efficient key assignment structure. The graphs show averages over 100 networks. The dotted lines show m_{Ave}, r_{Ave} and E_{Ave} for a tree that was rebuilt after every join operation, and are essentially the same graphs shown in Figure 8(b) for balanced trees. Note, however, that to obtain the m_{Ave}, r_{Ave} and E_{Ave} achieved by tree reconstruction, we would need to rebuild the key tree after each member join.

Figure 10(b) shows the efficiency improvements obtained by On-line VP3 over choosing the first available leaf, for m_{Ave} (top graph). r_{Ave} (middle graph) and E_{Ave} (bottom graph). The performance improvement of On-line VP3 over simply choosing the first available leaf increases with N, averaging 36% in all m_{Ave}, r_{Ave} and E_{Ave}, and coming close to 50% for some values.

Figure 11(a) shows a comparison between trees of degree 3 and 4. The trees of degree $d = 4$ perform, on average, better than trees of degree 4. In [25] the authors proved that the optimal tree degree in terms of GC transmissions for member deletion is equal to $d^* = 4$. We observe that, on average, minimum number of GC transmissions gives also smaller m_{Ave}, r_{Ave} and E_{Ave}. However, as can be seen, and unlike the wired case, a tree of degree $d = 4$ does not always outperform a tree of degree $d = 3$ in wireless networks.

Finally, we note that a strategy is required to determine at what point in the multicast group's lifetime, reconstruction of the key tree using VP3 will become a more cost effective option than insertion of joining members using On-line VP3. Rebuilding is a costly decision since, as Figure 11(b) shows, costs are at least an order of magnitude higher than E_{Ave} for most values of N. The strategy to decide between On-line VP3

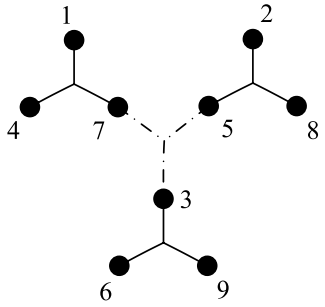


Fig. 12. Reduction of 3DM to MWN3PM. Assume that $W = \{1, 2, 3\}, X = \{4, 5, 6\}, Y = \{7, 8, 9\}$ and $U = \{(1, 4, 7), (2, 5, 8), (3, 6, 9), (3, 5, 7)\}$. Now we have $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, and we know $H(V, A)$ is a fully connected graph. The figure shows all vertices in V, and only those hyperedges in A that have weight 1, those hyperedges whose endpoint sets are elements of U. The minimum weight matching $\mathcal{M}'$ in our transformed hypergraph $H(V, A)$, is represented by those hyperedges shown in solid lines. The hyperedge shown in dotted lines is the only one of weight 1 that is not an element of $\mathcal{M}'$. Clearly, the only way $w_h(\mathcal{M}') = 3$, the cardinality of W, X and Y, is if there exists a match $\mathcal{M} \subseteq U$.

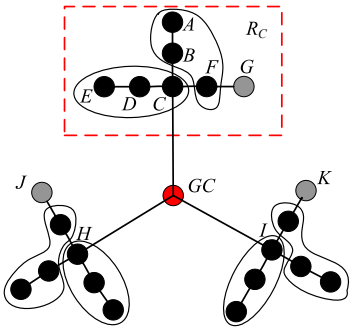


Fig. 13. (a) Worst case for VP3. For the subtree rooted at C, VP3 will select the following subgroups: $\{A, B, F\}, \{C, D, E\}$ first, and leave node G isolated. Similar choices will leave nodes K and J isolated. Therefore $S_7 = \{G, K, J\}$.

and reconstructing the key tree using VP3 would be based on user policies, and is beyond the scope of this paper.

VII. Conclusion

We addressed the problem of resource-efficient Key Distribution (KDP) for group communications in wireless ad-hoc networks. We considered the KDP under four metrics, namely member key storage, GC transmissions, MG update messages and average update energy. For each metric we formulated an optimization problem and showed that the KDP problem has unique solutions in terms of member key storage and GC transmissions, while it is NP-complete in terms of MG messages and average update energy. We proposed a cross-layer heuristic algorithm called VP3, that exploits network flows and algebraic structures, in order to assign KEKs to subgroups of members in an energy and bandwidth efficient way. We provided a worst-case bound for the grouping decision process of VP3 by introducing the notion of path divergence. We also proposed On-line VP3, a heuristic algorithm that inserts new members in the key assignment structure in an energy efficient way. Finally, we provided extensive simulations illustrating the savings achieved using our algorithms.

Appendix I
Minimum Weight Non k-Partite Matching Problem

Proposition 5: The Minimum Weight Non k-partite Matching problem is NP-hard for $k \geq 3$.

Definition 11: A *Hypergraph* $H(V, A)$ is a graph in which the set of generalized edges A, may connect more than two vertices in V. Edges in a hypergraph are commonly referred to as hyperedges.

Definition 12: A *k-uniform hypergraph* $H(V, A)$ is one in which all edges have k endpoints.

Proof: We begin by stating the 3DM problem:

Instance: A set $U \subseteq W \times X \times Y$, where W, X and Y are disjoint sets, and all have the same number of elements q.

Problem: Is there a subset $\mathcal{M} \subseteq U$, such that $|\mathcal{M}| = q$ and no two elements in $\mathcal{M}$ agree on an element from W, X or Y? That is, is there a matching $\mathcal{M}$ in U?

Now we state MWN3PM:

Instance: A 3-uniform hypergraph $H(V, A)$.

Problem: Find an optimal (minimum weight) matching $\mathcal{M}'$ in $H(V, A)$.

We first transform the input from 3DM by constructing a fully connected 3-uniform hypergraph $H(V, A)$, such that $V = W \cup X \cup Y$, where the hyperedge $a = (w, x, y)$ has weight $w_h(a) = 1$ if $(w, x, y) \in U$ and $w_h(a) = 2$ otherwise.

We then observe that because H is fully connected, there will always be an optimal matching $\mathcal{M}'$ in it. Next we note that $\mathcal{M}'$ will have one of two forms:

1) If there exists a match $\mathcal{M}$ that is the solution to the 3DM problem, the optimal match $\mathcal{M}'$ = MWN3PM$(H(V, A))$ will have weight $w_h(\mathcal{M}') = 3$, and $\mathcal{M} = \mathcal{M}'$.
2) If there is no match $\mathcal{M}$ in the original 3DM problem, then $w_h(\mathcal{M}') > 3$.

Thus, we have reduced the 3 Dimensional Matching problem to the Minimum Weight Non 3-Partite Matching problem. ■

Finally, we note that the generalized version of 3DM, the k-Dimensional Matching problem (kDM) is also NP hard, for all $k > 2$ [9], and that the reduction of kDM to MWNkPM, for all $k > 3$ is analogous to the one we have just shown.

Appendix II
Analytical Bound for Maximum Cumulative Path Divergence, $\Delta_d^*(S)$, for $d > 2$

Proposition 6: The maximum cumulative path divergence $\Delta_d^*(S)$, for a subgroup S of size $d > 2$ is:

$$\Delta_d^*(S) = (d-1) \max H_w(i), i \in R.$$

Proof: VP3 will achieve its worst-case bound when $|S \backslash \alpha(S)| = (d-1)$ subgroup members have $\Delta(\alpha(S), i) = \max[H_w(j), i \in S \backslash \alpha(S), j \in R]$. We present a construct

that achieves this worst-case bound in Figure 13. Assume $d = 3$ and, without loss of generality, assume $E_A > E_E > E_G > E_F > E_D$, so that VP3's first subgroup choices within the subtree R_C rooted at node C will be $S_1 = \{A, B, F\}$ and $S_2 = \{C, D, E\}$, as shown in Figure 13. Note that these choices leave node G isolated from those nodes of the broadcast routing tree R that have not been grouped yet. Similarly, VP3's subgrouping choices for subtrees R_H and R_I, rooted at nodes H and I respectively, will leave one isolated node in each subtree, nodes J and K. Since the roots of $d = 3$ subtrees are all connected to GC, the only subgrouping choice there remains for VP3 to take, is $S_7 = \{G, J, K\}$, the three nodes shown in gray in Figure 13. Note that the paths of the three subgroup members have a common node in GC, hence, the length of their common path is zero, and $\Delta(\alpha(S_7), i) = H_d(i, GC) = H_w(i), \forall i \in S_7 \backslash \alpha(S_7)$. Finally, since $H_w(G) = H_w(J) = H_w(K) = \max H_w(i)$ for $i \in R$, and $|S_7 \backslash \alpha(S_7)| = (d-1)$, we have that $\Delta(S_7) = (d-1) \max H_w(i)$. ■

REFERENCES

[1] M. Čagalj, J.P. Hubaux and C. Enz, "Minimum-Energy Broadcast in All-Wireless Networks: NP-Completness and Distribution Issues," in *Proc. of the ACM Mobile Communications Conference, MOBICOM 2002*, Atlanta, GA, 2002.

[2] R. Canetti, T. Malkin and K. Nissim, "Efficient Communication-Storage Tradeoffs for Multicast Encryption," in *Eurocrypt 99*, 1999.

[3] A. Clementi, P. Crescenzi, P. Penna, G. Rossi, and P. Vocca. "A Worst-case Analysis of a MST-based Heuristic to Construct Energy-efficient Broadcast Subtrees in Wireless Networks," TR 010, Univ. of Rome Tor Vergata, available at http://www.mat.uniroma2.it/ penna/papers /stacs01-TR.ps.gz, 2001.

[4] J. Edmonds, "Maximum Matching and a Polyhedron with (0,1) Vertices," in *Journal of Research of the National Bureau of Standards*, 69B, pp. 125-130, June 1965.

[5] L. Eschenauer and V. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," *In Proc. of ACM CCS*, 2002.

[6] A. Fiat, and M. Naor, "Broadcast Encryption," Lecture Notes in Computer Science, vol. 773, 1994.

[7] H. Gabow, "Data Structures for Weighted Matching and Nearest Common Ancestors with Linking," in *Proceedings of the 1st. Annual ACM SIAM Symposium on Discrete Algorithms (SODA '90)*, pp. 434-443, San Francisco, CA, USA, January 1990.

[8] J. Goshi and R.E. Ladner, "Algorithms for Dynamic Multicast Key Distribution Trees," in *Proc. of the Annual Symposium on Principles of Distributed Computing, PODC 2003*, 2003.

[9] R.M. Karp, "Reducibility Among Combinatorial Problems," in *Complexity of Computer Communications*, pp. 85-103, Plenum Press, New York 1972.

[10] E. Lawler, *Combinatorial Optimization: Networks and Matroids*, Dover Publications, February 2001.

[11] 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.

[12] L. Lazos and R. Poovendran, "Power Proximity Based Key Management for Secure Multicast in Ad Hoc Networks," to appear in *ACM Journal on Wireless Networks (WINET)*.

[13] L. Lazos, J. Salido and R. Poovendran, "VP3: Using Vertex Path and Power Proximity for Energy Efficient Key Distribution," in *Proc of IEEE VTC 2004*, September 2004, Los Angeles, USA.

[14] L. Lazos, R. Poovendran and G.H. Cirincione, "Location-Aware Secure Wireless Multicast in Ad-Hoc Networks under Heterogeneous Path-loss," in *University of Washington Electrical Engineering Technical Report Series, UWEETR-2003-0012*, Seattle, WA, 2003.

[15] X. Li, Y. Yang, M. Gouda and S. Lam, "Batch Re-Keying Approach for Secure Multicast," in *Proc. of the World Wide Web 10 (WWW10)*, Hong Kong, China, May 2001.

[16] M. Moyer, J. Rao and P. Rohatgi, "Maintaining Balanced Key Trees for Secure Multicast," Internet draft, draft-irtf-smug-key-tree-balance-00.txt, June 1999.

[17] R. Needham and M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *In Commuinications of the ACM*. 21(12): 993-999, 1978.

[18] R. Poovendran and J.S. Baras, "An Information-Theoretic Approach for Design and Analysis of Rooted-Tree-Based Multicast Key Management Schemes," *IEEE Transactions on Information Theory*, vol. 47, no. 7, November 2001.

[19] S. Setia, S. Koussih and S. Jajodia, "Kronos: A Scalable Group Re-Keying Approach for Secure Multicast," in *Proc. of the IEEE Security and Privacy Symposium 2000*, Oakland, CA, USA, May 2000.

[20] J. Snoeyink, S. Suri and G. Varghese, "A Lower Bound for Multicast Key Distribution," in *Proc. of IEEE Inforcom 2001*, 2001.

[21] Y. Sun, W. Trappe and K.J. Ray Liu, "A Scalable Multicast Key Management Scheme for Heterogeneous Wireless Networks," in *IEEE/ACM Transactions on Networking*, Vol. 12, No. 4, pp. 653-666, August 2004.

[22] Y. Vardi, "Metrics Useful in Network Tomography Studies," in *IEEE Signal Processing Letters*, Vol. 11, No. 3, pp. 353-355, March 2004.

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

[24] 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.

[25] 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.

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

[27] X.B. Zhang, S.S. Lam, D.Y. Lee and Y.R. Yang, "Protocol Design for Scalable and Reliable Group Rekeying," in *IEEE/ACM Transactions on Networking*," December 2003, Vol. 11, No. 6, pp. 908-921.

Javier Salido received his B.S. degree from the Universidad Autónoma Metropolitana in Mexico city in 1990, and his M.S. in Electrical Engineering from the University of Washington in 2005. He is currently working at Microsoft Corp.

Loukas Lazos received his Ph.D degree and M.S. degree from the same department in 2006 and 2002 respectively and his Engineering Diploma degree from the National Technical University of Athens, Greece, in 2000. His current research interests focus on cross-layer designs for resource-efficient network function for wireless ad-hoc networks, secure localization systems for sensor networks. Dr. Lazos is currently a post Doctorate at the Network Security Lab at the University of Washington.

Radha Poovendran is an Associate Professor at the Electrical Engineering Department of the University of Washington. He received his Ph.D. in Electrical Engineering from the University of Maryland, College Park in 1999. His research interests are in the areas of applied cryptography for multiuser environment, wireless networking, and applications of Information Theory to security. He is a recipient of the Faculty Early Career Award from the NSF CAREER (2001), ARO YIP (2002), ONR YIP (2004), and the PECASE (2005), for his research contributions.