Post Deployment Secure Key Management in Wireless Ad hoc Networks

Paul Loree and Kendall Nygard
Dept. of Computer Science
North Dakota State University
Fargo, ND USA
Email:{paul.loree, kendall.nygard}@ndsu.edu

Abstract— **Providing secure communication between nodes in mobile ad hoc networks is critical in many applications. In this paper, we present our work on a key distribution scheme for mobile ad hoc networks. Unlike traditional key distributions which establish encryption keys prior to node deployment we devise a post-deployment key distribution scheme to allow nodes a higher chance of connectivity with neighboring nodes. At the same time, nodes are able to be highly mobile within the network while reducing the need for excessive storage of encryption keys by each node in the network.**

Keywords-mobile ad hoc networks; wireless ad hoc networks; network security; key management; key distribution; secure communication; WANET; MANET; VANET.

I. INTRODUCTION

Wireless technology has advanced rapidly in recent years. New advances in low powered microcontrollers, such as Atmel [1] and ARM [2], and miniature computers [3] [4] have allowed for wireless enabled devices to be networked together to provide novel and interesting devices both commercially and by hobbyists. One of the promising networking techniques advanced in recent years are wireless ad hoc networks (WANETs). WANETs are organized as a decentralized network which operate over a wireless medium to provide communication services among different devices. Unlike an infrastructure network, WANETs are self-organizing, whereby the nodes in the network cooperate to provide the routing of data. One of the benefits of using wireless WANETs is robustness to node failure. Nodes participating in an ad hoc network join and leave the network randomly while not disrupting the network and its ability to route messages towards the destination. WANETs have several applications, such as in personal area networks (PANs), sensor networks (WSNs) vehicular WANETs (VANETs), and mobile WANETs (MANETs).

WANETs may also consist of nodes with different capabilities. For example, an ad hoc network may consist of several battery operated low powered devices, such as sensor nodes, and more powerful nodes, such as a laptop or tablet computer, which aggregate and process the sensor node data. MANETs and other WANETs have applications in many areas, including military surveillance, emergency incident response, industrial and agricultural monitoring, and in-home automation. For example, a military deployment may utilize a UAV-MBN network which consists of three levels of ad hoc networks operating together. At the lowest level the military may distribute sensor nodes into a battlefield environment which communicate information back to devices being used by troops on the ground through an ad hoc network [5]. The devices used by the troops are then connected through a localized mobile backbone network consisting of ground vehicles in the middle layer. At the highest level are unmanned aircraft which are used as a backbone to connect the ground vehicles over rough terrain and distances. Other uses of an ad hoc network include extending the range of existing infrastructure networks to allow additional devices to communicate via the Internet while out of range of infrastructure equipment, or connecting devices to the "Internet of Things". Securing the communication between the nodes in these networks is essential for high resistance to attackers who want to disrupt communication or gather information from the network.

The heterogeneous nature of these networks presents unique challenges for providing secure communication among all parties in the network. Resources available to devices found in the network may vary between devices and have limited capacity. Sensor network nodes, for example, may be limited by computational power, transmission range, and rely on battery resources for a power source. However, a laptop with an external wireless card connected to a power source does not have these limitations. This creates a serious problem for heterogeneous WANETs, as some devices may allow existing security techniques to be used, but they may not apply to other devices in the network.

To provide confidential communication in a network, key distribution must take place. A simple solution would be to preload a single key for encryption on each node. However, this approach has a drawback, because an attacker would only need to compromise one node in the network. Since WANETs rely on the cooperation and trust of all nodes in the network to handle routing, the compromised node is likely to receive a significant amount of data that may contain confidential information. Another simple solution using current technologies is to use a transient key based on a passphrase, as seen in 802.11 infrastructure networks. While this allows for easy implementation in the 802.11 protocol for securing the traffic on the network, not all WANETs may be able to utilize these security mechanisms. Additionally, nodes joining the network must know the predefined passphrase to connect to the network. This may not be possible in all situations, since the passphrase may not be known. For example, a VANET may consist of vehicles made by different manufacturers using their own passphrases for vehicles to form VANETs. This would reduce the usage of the VANETs being formed. To be effective in the unique environment of MANETs, some

 ISBN: 978-1-61208-461-9

method of encryption key distribution must be devised that does not rely on existing methods.

This paper is organized as follows: in Section II, we present related works in key management schemes for MANETs. Section III discusses our key distribution scheme. Section IV discusses our preliminary analysis and experimentation results of our proposed key distribution scheme. We conclude the paper in Section V and discuss our future plans for this research.

II. RELATED WORKS

Many key distribution schemes have been proposed in recent years for providing encryption keys to nodes in various types of WANETs. In this section, we discuss several of these proposed methods.

One of the first schemes proposed for establishing secure communication in WANETs was described in [6]. Eschenauer and Gligor's scheme was designed to be deployed in WSNs. Each node randomly selects a subset of keys from a key pool prior to being deployed in the network. After deployment, the nodes establish secure communication if two nodes share a common key. However, this scheme may not guarantee secure communication can be established since the scheme is based on the probability that nodes share a common key. To increase the probability a larger subset of keys must be preloaded which also increases vulnerabilities if a node is captured as more keys are exposed to the attacker. Chan et al. improved upon the security of this scheme in [7] by proposing that each node must have at least q keys in common between communicating nodes at a cost of additional memory.

In [8] and [9], two independent research studies proposed predistribution schemes which generate a set of key spaces and preload each node with a subset of the key spaces, known as keying shares. The difference in their approaches exist in the underlying mathematical framework used. In [8] the scheme uses a key space approach found in [10], while in [9] the authors generate key spaces based on the work found in [11]. Both allow communication to be secured between nodes if they share the same key space, but they rely on the probability of two nodes sharing keying information after deployment. This research expands on the use of keying shares as described in [10] and [11] to provide the generation of encryption keys after nodes have been deployed.

In [12], Hai-tao divides the network into zones based on the connectivity of nodes in a general area and a CH is selected to manage keys for each zone. The cluster key is created at the CH by nodes signing a nonce with their preloaded private key. Each of the nodes is then provided a shadow of the key and reconstructs the cluster key from the signatures of its neighbors.

PushpaLakshmi et al. proposed an agent based composite key management scheme in [13]. Nodes are partitioned into clusters in the network and a CH is selected based on its trustworthiness and probability the node will remain in the network. In their scheme, a fuzzy logic algorithm is used to determine the trustworthiness based on the node's successful routing statistics. Each node is also assigned a public certificate by an offline CA prior to joining the network and CHs are initially selected by an administrator. Additionally, a subset of nodes in the deployment will be used for creating partial private keys for new nodes to join the network. When a node joins the network the node generates its public key and registers with the CH that assigns a unique ID to the node. The CH then assembles the private key for the new node. The main drawbacks to this approach are that an offline CA is required, the initial CH nodes must be assigned by an administrator, and CHs know the private keys for joining nodes.

Liu et al. proposed an in situ key management scheme in [14] which establishes keys in homogeneous network after deployment. Their scheme elects nodes as CHs in a wireless sensor network to act as key distribution centers in the network based on a predetermined probability factor. When a node is elected, it generates a key space and distributes keying shares to nodes nearby which request keying information. However, their scheme was designed to be used in a homogeneous network and allowed the CHs to die after distributing the keying information.

In [15], Loree et al. proposed a similar scheme that extended on the work proposed in [14] by introducing a scheme designed to more efficiently provide keying information to a heterogeneous sensor network which used more powerful nodes to act as CHs. This work was further investigated in [16] and is the basis for this work in MANETs.

Zhao et al., [17] also proposed a scheme that uses a CA to provide public key encryption keys to wireless mesh networks. Unlike the previous scheme however, the CA is part of the infrastructure backbone of the network and nodes register when joining the network. Additionally, the CA must remain in the network which may not be suitable in a MANET environment.

Boukerche et al. proposed a key management scheme for MANETs in [18]. Their scheme provides both asymmetric and symmetric encryption to nodes in the network. Their work attempts to address the issue that a CA may not be present in the network to after deployment. They do require a CA to exist to distribute public/private keys to nodes prior to the network being formed. Additionally, each node is required to store both public keys and session keys of its neighbors. Lastly, since each message establishes a session between nodes, significant communication overhead occurs due to session key being established between nodes in the route.

In [19], Chauhan and Tapaswi describe a key management scheme that provides both asymmetric and symmetric encryption. In their scheme several nodes are preloaded with a public/private keys and a function to create key pairs for other nodes in the network. Each node stores either a symmetric key or a public key for use with each of the nodes in their cluster. Nodes are allowed move between clusters but must be validated by the previous CH before the node is given new keys. However, this is a drawback in this scheme when a node joins the network because the CH must know about the node prior to being added. Another drawback to this scheme is that nodes may only join clusters they are assigned to prior to setup or each CH would require information about all of the nodes in the network.

Lu et al., proposed a certificateless key distribution scheme in [20]. Their scheme distributes parts of a master

secret key to several key generating centers (KGC) that provide keying information to the nodes. In their approach a several KGCs must be contacted by a node to receive keying information, reducing the chance a single KGC is compromised the network if attacked. In their approach each node is able to calculate its neighbor's public key for secure communication with the neighbor's ID and the current key phase's salt value.

In [21], Dahshan and Irvine propose a MANET key management scheme which distributes threshold cryptography keys. In their scheme the nodes in the network consist of two types of devices, CA trusted nodes authenticated prior to deployment and non-trusted nodes which joined after deployment. Nodes establish a connection-orientated route to the destination only through CA trusted nodes. Both source and destination nodes may then either accept or deny the route to be used. Nodes are also able to individually revoke certificates if they believe it has been compromised. However, each node must keep a public certificate for all nodes in a route which may not scale well especially if nodes are required to persistently store all received certificates. If nodes are allowed to delete the certificates there will be significant communication overhead when establishing a route for each message and nodes would no longer be able to revoke keys. Lastly, only CA authenticated nodes are used in a route and the network may not be fully connected since nodes are mobile and may have moved out of range of CA authenticated nodes. In order to ensure that nodes stay connected in their scheme enough statically located CA authenticated nodes must be placed knowingly in the topology of the network prior to deployment.

In [22], Seghal et al. presented analysis of security issues in MANETs. Their work provided analysis of the existing problems facing three key areas of MANETs, key management, ad hoc routing, and intrusion detection. They state two main problems are faced when dealing with key management in MANETs. The first problem is it is difficult for nodes in the network to determine if nodes in the network have revoked a certificate used by a node in the network. A second problem is that nodes may be in different trust hierarchies and their certificates may not be valid across different levels in the network. They propose the solution to this is to use a trusted third party (3P) or global password authentication for all nodes to use to gain access. Both of these however pose drawbacks since MANETs may be created in places with limited infrastructure or a 3P CA would be infeasible. Global passwords also provide limited security since clients with malicious intent can gain access to this information easily. This demonstrates the need for a distributed key distribution system that can be efficient to deploy to nodes joining the network without compromising the security of existing or future clients in the network.

III. WANET KEY DISTRIBUTION SCHEME

In this section, we describe our key management scheme for providing keys for symmetric encryption to nodes in a WANET. We expand on our work in [15] and [16] for support of WANETs. In our scheme nodes are partitioned into clusters and CHs are elected post-deployment to provide keying information for nodes to generate encryption keys. A secret key is generated between two nodes using key space models discussed in [11] and [10]. These key spaces are generated using either a bivariate symmetric n-degree polynomial [11] or symmetric public and private matrices of $(n + 1) \times (n + 1)$ dimensions [10]. The coefficients of the polynomial or the elements of the matrices are generated after the network is deployed. Keying information is distributed to the nodes in a cluster by a CH. By using the keying information, each node is able to create secret keys for symmetric encryption between themselves and their neighbors using their IDs as input.

A. Key Space Models

Our scheme will operate under both key space models described in [11] and [10]. In [11], the authors utilize a symmetric n-degree polynomial. In the first key space model we use the bivariate symmetric n-degree polynomial such that

$$f(x, y) = f(y, x) = \sum_{i,j=0}^{n} a_{i,j} x^j y^j \qquad (1)$$

over a finite field F_r, where r is a large prime number that can be used for cryptographic keys. A key for symmetric encryption can be found by two nodes in the network that share the same coefficients, a_i, the key space, by exchanging IDs and computing (1). In our scheme, the CH generates a set of functions, F, and each node in the cluster is provided with one of the generated functions as its keying information. When creating a secure link between two nodes in the cluster the nodes send their calculated coefficients for their function using their ID, x. This allows the receiving node to compute the same key by inputting their ID, y to create the secret key.

In [10], the authors proposed a similar method that uses matrices instead of a polynomial. Nodes exchange column values of a public matrix along with their ID to compute a key. One benefit to using this key space method is that communication overhead can be reduced by using a generation matrix, e.g. Vondermonde matrix [8], where a seed value is used to compute a column. This however increases the computational overhead by $n - 1$ modular multiplications.

A valuable property in both of these models is that they are both n-collusion resistant. That is, where less than n nodes using the key space remain uncompromised, the key space itself remains secure and new keys generated by the key space cannot be determined using the information contained in the compromised nodes as shown in [11] and [10]. Furthermore, both of these models have low storage overhead as shown in [14].

B. Assumptions

In our work, we assume no prior knowledge of node placement before deployment. We also assume that nodes are mobile and may move between clusters in the network. Nodes may belong to several clusters if they are within transmission range of a CH. CHs are also assumed to operate normally in the network in addition to CH responsibilities. We also

 ISBN: 978-1-61208-461-9

assume each node is loosely time synchronized within the network. Furthermore, the network supports an ad hoc routing protocol such as Ad hoc On-demand Distance Vector (AODV) or Dynamic Source Routing (DSR). Each node is preloaded with various parameters prior to joining the network described in Table 1.

TABLE I. PRELOADED PARAMETERS

λ	Maximum number of nodes in a cluster
n	Degree of polynomial or matrix dimensions for generating key spaces
ID	Unique ID
r	Large prime for generating key space over finite field
t_{wait}	Maximum bootstrapping wait time
$t_{threshold}$	Minimum time period before rekeying for new node joins
t_{limit}	Time period for each key share in network
TTL	Time to Live
p, q	Large primes needed for Rabin's cryptosystem
B	Predefined padding for Rabin's cryptosystem

Each cluster has a maximum of λ nodes. Since the key space models are n-collusion resistant the optimal value for the cluster size should be less than or equal to the value of n. Each node contains a unique ID and large prime number, r, to be used to generate key spaces over a finite field, F_r. Each node is also preloaded with three threshold values. These thresholds are used to define the maximum wait time, t_{wait}, before a CH announces it has generated keying information, the minimum amount of time a node is allowed to join before rekeying, $t_{threshold}$, and a key expiration time limit, t_{limit}. Additionally, the maximum time to live for packets being broadcast from the CH is defined by TTL. The last three values are optional if the network is assumed to be insecure at startup then these parameters are used for Rabin's cryptosystem as described in the next section.

C. Rabin's Cryptosystem

In our scheme we assume that the network may be insecure from eavesdropping. In this network model we use Rabin's Cryptosystem [23] to temporarily secure the transmissions between the CH and joining nodes while distributing keying information. This allows new nodes to join without requiring a CA to preload public/private key pairs to new nodes joining the network.

Rabin's cryptosystem is a computationally asymmetric encryption algorithm that uses two large prime numbers to create a public key $n = pq$. The CH sends n to cluster nodes to encrypt a temporary session key, k. The nodes in the network randomly generate k to encrypt the exchange of keying information. Using Rabin's algorithm this session key is encrypted as $E_n(k||B) = (k||B)^2 \ mod \ n$, where B is a preloaded bit pattern. B is required to allow the CH to decrypt the message successfully using Rabin's algorithm. In addition to the session key the node also sends its ID to the CH. We use this cryptosystem because it is computationally cheap for encryption but as computationally expensive as RSA to factor the two large prime numbers, p and q from n.

D. Cluster Head and Key Distribution Algorithms

In this subsection, we describe two algorithms used by CHs to generate key spaces and distribute keying information to nodes in the cluster. The first algorithm we discuss is the Cluster Head Algorithm illustrated below in Figure 1.

In our scheme we elect CHs using a predefined probability threshold value. An alternative would be to elect nodes based on their performance capabilities in the network. If performance capabilities are measured, the CHs can be selected based on several metrics such as computational performance, battery life, wireless medium interfaces on the device, number of neighboring nodes within a specific hop distance, or transmission range.

Once elected, the CH generates a key space based on one of the key space models previously described on Line 2. On line 3, the CH generates a random wait time, w less than the preloaded maximum wait time, t_{wait}. The CH then listens for other nodes that may have also elected themselves to be the CH for an area in the network as shown in the loop starting on line 4. This is to reduce collisions between two nodes in an area of the network that may be simultaneously broadcasting an announcement that they have keying information available and to let the first node announcing it has keying information available to become the CH. Once the announcement has been broadcasted the CH starts the distribution process as illustrated below in Figure 2.

Figure 2 below describes our Key Distribution Algorithm. On line 2, the CH announces its ID to the network to notify nodes within TTL hops that keying information is available. After sending the announcement the CH will start a loop on Line 3 and accepts requests from other nodes in the network within range of its broadcast for keying information. Nodes send requests to the CH using unicast messages. This loop will continue to provide keying information as long as the cluster size limit, $\lambda - 1$, has not been reached. The CH will also save one keying information share for itself so it remains in the cluster it has formed. When the CH receives a request for keying information the CH selects an unused keying information share as shown on Line 6. The CH will also mark

```
Cluster Head Algorithm
 1: function runClusterHead (λ, t_w)
 2:     keys ← genKeySpace()  // construct key space forλ nodes,
                                 store one for self
 3:     w ← random(0 < t_w) // random wait time
 4:     while w > 0 do
 5:         listen for broadcasts from neighboring nodes that
            have elected themselves as cluster heads
 6:         if BROADCAST heard
 7:             RequestKeyInfo() // get keying info
 8:             exit
 9:         end if
10:         elapse(w)
11:     end while
12:     KeyDistro() // Fig. 2
13: end function
```

Figure 1. Generates Keying Information

```
Key Distribution Algorithm
 1: function KeyDistro(λ, ID, keys)
 2:     BROADCAST(ID) // broadcast to all nodes within TTL hops
 3:     keyed := 1
 4:     while keyed < λ − 1 do
 5:         if received(RequestKeyInfo(ID_k))
 6:             KeyingInfo ← an unused key share
 7:             usedKeys ← usedKeys ∪ {KeyingInfo}
 8:             send(ID, k(KeyingInfo)) // k if using Rabin's
 9:             keyed := keyed + 1
10:         end if
11:     end while
12: end function
```

Figure 2. Distributes Keying Information

the used keying information so that it will not provide this information to another node in the cluster as shown on Line 7. On Line 8, the CH will send an unused keying information to the requesting node using a unicast message. The distribution algorithm runs until all keying information has been used. As long as keying information remains available the cluster may accept new nodes joining the network.

E. Cluster Joining Algorithm

In this subsection, we describe our algorithm for forming clusters and nodes joining a cluster in the network after the network has been established.

Figure 3 shows our Cluster Joining Algorithm. When a node joins a cluster it requests the remaining time until the rekeying phase, as shown on Line 3. Only the CH will respond to this request with a broadcast, so existing nodes are informed of the new node and the remaining time before rekeying the cluster begins. If there is no response, then a cluster does not exist and the node will begin the election process to become a CH, as previously described in Figure 1 and the function exits. If there is a response from a CH and the $t_{remaining} \leq t_{threshold}$ the re-election and rekeying event will be started early and the node will join the newly formed cluster during this phase. During the re-election process nodes will generate a probability or benchmark score on Line 9 and announce their score to their area in the network. The node with the highest value will then become the new CH for the cluster starting on Line 12. Once a node has been elected as a CH it will start generating key spaces as shown on Line 20. Nodes not elected will then request keying information once a CH broadcasts an announcement as described in the previous subsection. Nodes within range of more than one CH will also request keying information from each of the CHs it receives a broadcast from. This allows nodes which are lying on the fridges of a cluster to join multiple clusters and provide routing paths between clusters.

If the value of $t_{remaining} > t_{threshold}$ then the node joining the cluster will request the keying information from the current CH as shown on Line 25. Security can be improved by setting $t_{threshold} = t_{limit}$ since a new node will force rekeying. However, if the network is expected to have a lot of nodes moving into and out of the cluster than this may

```
Join Cluster Algorithm
 1: function joinCluster (t_threshold)
 2:     t_remaining := 0
 3:     t_remaining := RequestKeyTime ( ) // get remaining time of current key time from current cluster head
 4:     if(t_remaining == 0)
 5:         runClusterHead(λ, t_w) //initialization of network
 6:         exit
 7:     end if
 8:     if(t_remaining <= t_threshold)
 9:         score_new ←calcScore( ) // score node capabilities (probability, cpu, storage, batt, RSSI, node connectivity)
10:         chScore{} ← scoreRequest( ) // Broadcast request for cluster node scores
11:         BROADCAST score
12:         foreach score in chScore
13:             if(score_new > score)
14:                 newClusterHead ← true // elect self as new cluster head
15:             else
16:                 newClusterHead ← false // remove election if another node is better
17:             end if
18:         end foreach
19:         if(newClusterHead)
20:             runClusterHead( ) // become cluster head and generate keying information (Fig. 1)
21:         else
22:             RequestKeyInfo( ) // request keying information from new cluster head
23:         end if
24:     else
25:         RequestKeyInfo( ) // request keying information from existing cluster head
26:     end if
27: end function
```

Figure 3. Joins or forms network clusters

cause unwanted computational and communication overhead.

Additionally, the CHs will restart the keying process once the t_{limit} threshold value has been met. At this time either the node may generate new keys for their cluster or restart the election process to see if a new CH is available. To reduce the load on the previous CH the current CH may also be set to not elect itself again for a period of time and a round robin approach may be used to elect new CHs.

IV. KEY DISTRIBUTION ANALYSIS

In this section, we provide a performance analysis of our scheme. First we re-examine the security performance of the scheme. As shown in [10] and [11], both key space models require at least n nodes in a cluster to be compromised before all of the generated keys for symmetric encryption used in the key space can be calculated. We assert that to increase the security of these key space models that the degree should be at least $n \geq \lambda$. If the number of nodes in each cluster is the same as the degree of the polynomial or the size of the

matrices, this will ensure that only if all nodes in a cluster are compromised would the key space be compromised. At this point the attacker would already have access to all keys within the cluster. If $n > \lambda$, then determining any other keys generated in the future from the key space would be impossible, which keeps the future cluster nodes secure. Additionally, this will allow new nodes to join existing clusters before the rekeying process to starts or until all key shares have been distributed.

Next we consider storage requirements for nodes in the network for storing keys for symmetric encryption. Each node will be required to store a key for itself and any node it has received IDs from within its transmission range. If a node has joined k clusters then these nodes would be required to store at most $k\lambda$ keys for symmetric schemes. However, we expect most nodes in the network to only store the nodes within their immediate transmission range and only be connected to one cluster at a time except for nodes along the edges of clusters which join adjacent clusters to allow for communication between clusters. Additionally, nodes will store keying information for the key space. In [14], it is shown that storage requirements for keying information are close to $(n + 1)\log r$ in the polynomial based model and $(n + 2)\log r$ in the matrix model if the matrix selected for G allows the columns to be seeded.

Finally, we discuss the communication overhead of our scheme. In the following equations let N be the number of nodes in the network, C be the number of elected CHs and λ the number of nodes in each cluster. Below in (2) we show the total number of CH announcement broadcast messages transmitted by CHs in the network during the distribution phases of the scheme. In a network consisting of C clusters each elected CH, E_i, sends a broadcast message to their neighboring nodes. This announcement broadcast message is then retransmitted by at most λ nodes in each cluster while the $TTL > 0$.

$$Broadcasts = \lambda \cdot \sum_{i=1}^{C} E_i \qquad (2)$$

Once the announcements from the CHs have been transmitted to nodes within TTL range of elected CHs, each node within range joins the CH cluster. The total number of messages needed to distribute this information is be shown below in the (3). In order to join the cluster each node which receives the announcement replies by sending a unicast message requesting a keying information from the key space generated by the CH. This requires each node, R_j, up to λ nodes in a cluster to send a unicast message back to the CH. Each unicast message sent is routed up to TTL times in order to reach the CH. This process occurs in each of the C clusters in the network. Each CH then replies to up to λ nodes with a unicast message to each of the requests for keying information nodes.

$$Distributions = 2C \cdot \sum_{j=1}^{\lambda} (R_j + TTL) \qquad (3)$$

During the election and rekeying process each node in each cluster broadcasts its probability or benchmark score in order for the new CH to be selected. The total number of messages this requires is shown in (4) below. Each node in the network, S_k, broadcasts its score value which is then retransmitted a maximum TTL times by its neighbors.

$$Scores = \sum_{k=1}^{N} (S_k + TTL) \qquad (4)$$

The average number of messages sent per each node in our scheme can be calculated by (5) below which is the sum of the previous totals divided by the number of nodes in the network, N.

$$Average = \frac{Broadcasts + Scores + Distributions}{N} \qquad (5)$$

Limiting the average number of messages transmitted over the network by each node is important to increase the life expectancy of nodes relying on battery power.

V. CONCLUSION

We devised and analyzed an efficient secure key distribution scheme for WANETs. Our proposed scheme has strong security against attackers attempting to discover the shared secret keys used for symmetric encryption by being n-collusion resistant, which requires attackers to compromise an entire cluster of nodes before keying information can be discovered. We also provide efficiency in our scheme in terms of computational, storage and communication overhead in order to increase the life expectancy of battery powered nodes placed in the network.

Unlike existing schemes, our method does not rely on prior knowledge of the network topology and can establish keys for symmetric schemes to be used between nodes after deployment. Additionally, our scheme does not require centrally administered CAs to be used in the network to distribute keys to nodes prior to the nodes joining the network. Furthermore, our scheme allows all nodes to be mobile in the network and does not require nodes to remain statically assigned to locations in the network for handling routing and key distribution, which may be difficult in many situations, such as in MANETs.

In the future, we plan to empirically evaluate our scheme by using simulations to compare our scheme with competing existing schemes for connectivity between nodes. Additionally, we plan to investigate adding authenticity and integrity mechanisms into the scheme to improve the security of the network not covered under confidentiality provided by encrypted communication. We also plan to simulate our scheme against these other methods while under various attacks unique to WANETs such as Sybil, black hole, and wormhole attacks. Finally, we plan to compare our scheme against other methods for routing messages across the network between mobile nodes while under normal network conditions and while under attack.

REFERENCES

[1] "Atmel Corporation," [Online]. Available: http://www.atmel.com/. [Accessed November 2015].

[2] "ARM," [Online]. Available: http://arm.com/. [Accessed November 2015].

[3] "Arduino," [Online]. Available: http://www.arduino.cc/. [Accessed November 2015].

[4] "Raspberry Pi," [Online]. Available: http://www.raspberrypi.org/. [Accessed November 2015].

[5] H. Deng, R. Xu, J. Li, F. Zhang, R. Levy, and W. Lee, "Agent-based Cooperative Anomaly Detection for Wireless Ad Hoc Networks," in 12th International Conference on Parallel and Distributed Systems, 2006, pp. 1-8.

[6] L. Eschenauer and V. D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," in 9th ACM Conf. Computer and Communications Security, Washington, DC, 2002, pp. 41-47.

[7] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," in IEEE Symposium on Security and Privacy, Berkeley, CA, 2003, pp. 1-17.

[8] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney, "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks," in 10th ACM Conf. Computer and Communications Security, Washington, DC, 2003, pp. 42-51.

[9] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," in 10th ACM Conf. Computer and Communications Security, Washington, DC, 2003, pp. 52-61.

[10] R. Blom, "An Optimal Class of Symmetric Key Generation," in Conference on the Theory and Applications of Cryptographic Techniques, Paris, Fr, 1984, pp. 231-236.

[11] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-Secure Key Distribution for Dynamic Conferences," in 12th Annual International Cryptology Conference on Advances in Cryptology, Santa Barbara, CA, 1992, pp. 471-486.

[12] X. Hai-tao, "A Cluster-Based Key Management Scheme for MANET," in 3rd Int'l Workshop on Intelligent Systems and Applications, Wuhan, China, 2011, pp. 1-4.

[13] R. PushpalLakshmi, R. Rahul, and A. Kumar, "Mobile Agent Based Composite Key Management Scheme for MANET," in Int'l Conf. on Emerging Trends in Electrical and Computer Technology, Nagercoil, India, 2011, pp. 964-969.

[14] F. Liu, X. Cheng, L. Ma, and K. Xing, "SBK: A Self-configuring Framework for Bootstrapping Keys in Sensor Networks," IEEE Transactions on Mobile Computing, vol. 7, no. 7, July 2008 pp 1-11.

[15] P. Loree, K. Nygard, and X. Du, "Efficient Post-Deployment Key Establishment Scheme for Heterogeneous Sensor Networks," in IEEE GLOBECOM, Honolulu, HI, 2009, pp. 1-6.

[16] P. Loree, "Post-deployment Key Management in Heterogeneous Sensor Networks," North Dakota State University, Fargo, ND, 2010, pp. 1-63.

[17] X. Zhao, Y. Lv, T. H. Yeap, and B. Hou, "A Novel Authentication and Key Agreement Scheme for Wireless Mesh Networks," in IEEE 5th Int'l Joint Conf. on INC, IMS, and IDC, 2009, pp. 471-474.

[18] A. Boukerche, Y. Ren, and S. Samarah, "A Secure Key Management Scheme for Wireless and Mobile Ad Hoc Networks Using Frequency-Based Approach: Proof and Correctness," in IEEE Global Telecommunications Conf., 2008, pp. 1-5.

[19] K. K. Chauhan and S. Tapaswi, "A Secure Key Management System in Group Structured Mobile Ad hoc Networks," in IEEE Int'l Conf. on Wireless Communications, Networking and Information Security, Beijing, China, 2010, pp. 307-311.

[20] L. Lu, Z. Wang, W. Liu, and Y. Wang, "A Certificateless Key Management Scheme in Mobile Ad Hoc Networks," in 7th Int'l Conf. on Wireless Communications, Networking and Mobile Computing, Wuhan, China, 2011, pp. 1-4.

[21] H. Dahshan and J. Irvine, "A Trust Based Threshold Cryptography Key Management for Mobile Ad Hoc Networks," in IEEE 70th Vehicular Technology Conf., 2009, pp. 1-5.

[22] U. Sehgal, K. Kaur, and P. Kumar, "Security in Vehicular Ad-hoc Networks," in 2nd International Conference on Computer and Electrical Engineering, 2009, pp. 485-488.

[23] M. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization," Cambridge, MA, 1979, pp. 1-20.

 ISBN: 978-1-61208-461-9