

Efficient Hybrid Security Mechanisms for Heterogeneous Sensor Networks

Patrick Traynor, Raju Kumar, Heesook Choi,
Guohong Cao, Sencun Zhu and Thomas La Porta
Networking and Security Research Center
Pennsylvania State University
University Park, PA 16802

Abstract

Many applications that make use of sensor networks require secure communication. Because asymmetric-key solutions are difficult to implement in such a resource-constrained environment, symmetric-key methods coupled with a priori key distribution schemes have been proposed to achieve the goals of data secrecy and integrity. These approaches typically assume that all nodes are similar in terms of capabilities, and hence deploy the same number of keys in all sensors in a network to provide the aforementioned protections. In this paper, we demonstrate that a probabilistic unbalanced distribution of keys throughout the network that leverages the existence of a small percentage of more capable sensor nodes can not only provide an equal level of security but also reduce the consequences of node compromise. To fully characterize the effects of the unbalanced key management system, we develop, implement and measure the performance of a complimentary suite of key establishment protocols known as LIGER. Using their pre-deployed keys, nodes operating in isolation from external networks can securely and efficiently establish keys with each other. Should resources such as a backhaul link to a key distribution center (KDC) become available, networks implementing LIGER automatically incorporate and benefit from such facilities. Detailed experiments demonstrate that the unbalanced distribution in combination with the multi-modal LIGER suite offers a robust and practical solution to the security needs in sensor networks.

1 Introduction

The deployment of wireless sensor networks is becoming more common in a wide range of environments. In scenarios ranging from the remote observation of wildlife and the monitoring of so-called “smart” buildings to commercial inventory management and vehicle/target tracking, sensor networks are being employed for the task of distributed information accumulation. These systems have traditionally been composed of a large number (hundreds to a few thousand [1]) of homogeneous nodes with extreme resource constraints. This combination of austere capabilities and physical exposure make security in sensor networks an extremely difficult problem.

Because traditional asymmetric encryption is not practical in this environment, a number of clever symmetric-key management schemes have been introduced. One well received solution that has been extended by several researchers is to distribute a certain number of randomly selected keys in each of the nodes throughout the network [9, 4, 7, 15, 33]. Using this scheme, one can achieve a known probability of connectivity within a network. These previous efforts have assumed a deployment of homogeneous nodes, and have therefore suggested a balanced distribution of random keys to each of the nodes to achieve security. Likewise, the analysis of those solutions relies on assumptions specific to a homogeneous environment.

A deviation from the homogeneous system model has been increasingly discussed in the research community. Instead of assuming that sensor networks are comprised entirely of low-ability nodes, a number of authors have started exploring the idea of deploying a heterogeneous mix of platforms and harnessing the available “microservers” for a variety of needs. For example, Mhatre, et al. [18] automatically designates nodes with greater inherent capabilities and energy as cluster heads in order to maximize network lifetime. Traynor, et al [29] extend this idea to mobile groups by having a more powerful node perform group handoffs for neighboring sensors.

Heterogeneity also offers new possibilities in terms of network connectivity. The currently available schemes for connectivity in homogeneously composed sensor networks assume one of two scenarios: either a network with access to a KDC via a base station, or a remotely located, stand-alone system without access to any infrastructure. We argue, however, that the expansion of

heterogeneous networks allows for systems to potentially operate in a multi-modal fashion. For example, consider an air-deployed sensor network being dispersed over a disputed border area. Nodes arrive long before friendly forces are likely to be moving through the region and must therefore begin the process of secure data collection without help from external mechanisms. If nodes are equipped with symmetric keys through an a priori distribution scheme, neighbors will be able to exchange encrypted transmissions; however, it is extremely difficult for nodes to truly, scalably authenticate each other. If, at a later time, the arrival of allied troops provides access to a backbone network, a truly robust system should be able to harness the additional security guarantees such as centralized authority and scalable authentication provided by a KDC.

In this paper, we leverage heterogeneity to provide more robust key management and establishment protocols for sensor networks. First, we introduce the probabilistic unbalanced key management scheme. Unlike previous homogeneous key distribution schemes, the number keys stored by each node in the network is proportional to its inherent resources. We then derive probabilities for connectivity under a number of new key establishment trust models. Third, we design and implement a suite of protocols which we call LIGER¹. Networks running the LIGER suite can not only establish keys between nodes in absence of a backhaul connection, but also take advantage of such resources when they become available. This is the first proposed multi-modal key establishment suggested for sensor networks. Lastly, we characterize this hybrid protocol through an extensive performance analysis and examination of the savings gained from code re-use. In so doing, we demonstrate that the use of heterogeneity through the unbalanced key distribution and LIGER suite provides robust, efficient and practical mechanisms to support the secure operation of sensor networks.

The remainder of this paper is organized as follows: Section 2 examines pertinent related work; Section 3 details our new network model and explores the theoretical connectivity of such systems under a variety of key establishment trust models; Section 4 presents the specifics of LIGER, our hybrid key establishment protocol; Section 5 provides a detailed analysis of the performance and

¹A Liger is the hybrid offspring of a lion and a tiger.

overhead of our scheme compared to traditional, balanced approaches; Section 6 offers additional observations and discussion about the advantages and tradeoffs of such a scheme; Section 7 provides concluding remarks and discusses future work.

2 Related Work

The preceding work in the field of key management for wireless sensor networks can be broken into two categories: schemes that have constant access to a KDC or trusted third-party keying mechanism, and those that never do. While there has also been a large body of work on distributed certificate authorities applied to ad hoc networks, these schemes rely on public key cryptography and are therefore not directly applicable to sensor networks.

The wired world of networking is already familiar with a number of protocols for authentication and session key establishment. The classic protocol for authenticating communications between two machines was written by Needham and Schroeder [20]. Improvements to this protocol were made in the abundantly used Kerberos [13]. Fox and Gribble proposed the use of Charon [10], a proxy server designed to offload the memory overhead of Kerberos for mobile devices. However, as these protocols require a large number of transmissions in order to establish a single session key with neighbors, they are therefore not particularly well suited for this environment.

A more appropriate centralized keying method for sensor networks is proposed in the SPINS protocol [23], which makes use of a modified version of the TESLA [22] authentication protocol (μ Tesla). Each node in a network running SPINS contains only a pair-wise key with the base station/KDC and uses one-way hash chains for creating an epoch-delayed key release mechanism for use in authenticated broadcast. If two nodes A and B wish to establish keys with each other, A sends a request to B , which creates and forwards a token to the base station. The base station generates a session key, encrypts it in the secret keys that it shares with each of the involved parties and transmits the data. While this scheme has many attractive attributes, it will not operate if the base station/KDC is unreachable.

A large number of key distribution schemes have been proposed for networks that are unable to access a KDC after deployment. The most famous of these pre-distribution schemes is the

work by Eschenauer and Gligor [9]. Given a key pool of size P , nodes are preloaded with k keys (selected without replacement) such that two randomly picked nodes can communicate with a given probability (i.e., share at least one key). In order to determine whether or not a key is shared, each node broadcasts its key identifiers (which are randomly associated with the keys themselves before deployment) in plaintext. Neighbors sharing a key associated with one of those identifiers then issue a challenge/response to the source. If two nodes do not share keys directly, they can establish a session key with the help of neighbors with which a key is already shared. While this technique is well suited for establishing session keys in a stand-alone network, it does not provide support for authentication.

This work was expanded via a number of methods by Chan, et al [4]. One extension requires nodes to share at least q keys to establish secure communication links. This greatly reduces the possibility of an intruder being able to eavesdrop on communications through the compromise of a small number of peer nodes. Additionally, the authors suggest a distribution model in which each node stores pairwise keys between some subset of the nodes in the network. This allows nodes to authenticate peers with which they share one of the pairwise keys, and limits the damage done to uncompromised nodes when keys are exposed to an adversary.

Other authors propose different mechanisms for storing or generating keys to reduce memory requirements. In Zhu, et al. [32], the authors propose an erasure-based pre-deployment method to create secure channels over which group keys are distributed. A number of other researchers have proposed incorporating location-based information into the assignment of keys [8, 15, 16]. Because of the expensive and high power-drain characteristics of GPS units, these schemes typically rely upon initial node placement for this information. The use of polynomials [7, 6] and channel diversity [19] for establishing keys have also been investigated.

None of the previous work has fully defined a protocol or directly measured the performance of an implementation of the security methods they propose. The implementation and measurements presented here show the viability of these systems in a practical environment and illustrate their sensitivity to MAC layer collisions and network density.

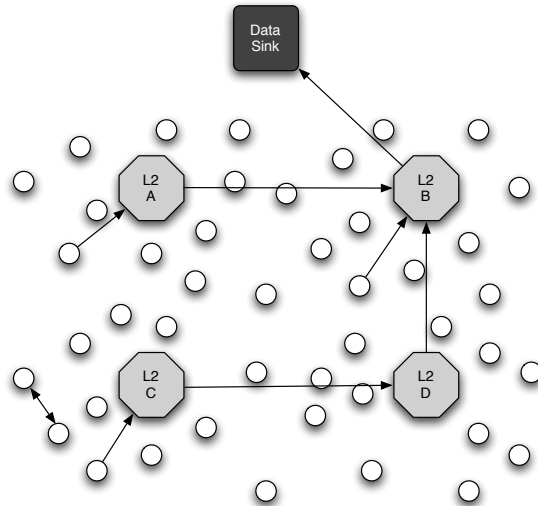


Figure 1: New models for connectivity available in heterogeneous sensor networks. Sensors can communicate with their neighbors in the traditional means, or act as neighborhoods in a larger ad hoc network. A variety of radios also offers direct or indirect access to external resources.

3 Network Model

This section begins by exploring a new network model made possible by heterogeneity. We then introduce the unbalanced key distribution and a series of trust models for key establishment. The goal of these protocols is to create secure links between neighboring source and destination nodes, thereby providing secure hop-by-hop communications across the network. In so doing, this solution for key distribution allows systems to operate securely while minimizing the burden placed upon the least capable nodes in the system.

3.1 Heterogeneous Sensor Networks

The previous work on random key pre-deployment in sensor networks has assumed either a grid or very large random-graph arrangement such that all neighbors within the transmission radius of a given node are reachable. Communication between adjacent nodes is therefore limited only by key matching. This model is not always realistic for a number of reasons. Primarily, it fails to take in to account that signal-blocking barriers including hills, buildings and walls may exist between neighboring nodes. More importantly, it fails to consider that better radio technology and resources are available to some members of the network. Because the assumptions of topology and topography used in most previous approaches are violated in realistic settings, we propose a new network model.

Figure 1 shows a network model with two main differences from that used in previous work. First, the landscape over which a sensor network is placed contains features that may segregate nodes into exclusive neighborhoods. Because nodes are still distributed through the same methods discussed in previous papers (e.g. dropped from an airplane), there is no way to determine a node’s neighbors a priori. Due to the random nature of the deployment, the potential for node mobility and addition of nodes at a later time, assigning keys for specific neighbors is not possible. Second, instead of a homogeneous composition of nodes, the network now consists of a mix of nodes with different capabilities and missions. The sensing or Level 1 (L1) nodes are assumed to be very limited in terms of memory and processing capability, and perform the task of data collection. These nodes are indicated as white circles in Figure 1. Level 2 (L2) nodes have more memory, processing ability and improved radios (e.g. 802.11). These nodes are equipped with additional keys, and take on the role of routers and gateways between networks. In addition to tamper-resistant casings [12], L2 nodes are assumed to be equipped with a fast encryption/deletion algorithm to protect their supplementary keys from compromise if they are captured. We discuss the relaxation of this assumption and examine the effects of L2 compromise in Section 6.

Such a model presents a number of new possibilities for sensor networks. First, many of the techniques that have proven successful in ad hoc networks (e.g. secure routing [3, 11, 21, 24]) can now be incorporated into sensor networks. Secondly, with at least intermittent connectivity to external networks, resources previously unavailable to sensor networks can now assist in their operation.

3.2 Unbalanced Key Distribution

Our scheme for the unbalanced distribution of keys throughout a wireless sensor network builds upon the previously described balanced approach of Eschenauer, et al [9]. Under the balanced approach, a large pool of P keys is generated, from which k are randomly selected, without replacement, for each sensor node. Two nodes may communicate to directly establish a session key if they have a key match. The probability that two nodes with the same number of random keys, k , share at least one key is:

$$P[Match] = 1 - \frac{((P - k)!)^2}{P!(P - 2k)!} \quad [9] \quad (1)$$

If nodes do not have a key match, they may still establish a session key through one or more intermediate nodes with which they each have a common key.

Given the same generated key pool of size P , we store a key ring of size k keys in each sensor (L1) node, and a key ring of size m keys in each L2 node, where $m \gg k$. The equation for the probability of an L2 and L1 having at least one key in common is given by Eq. 2.

$$P[Match] = 1 - \frac{(P - k)!(P - m)!}{P!(P - m - k)!} \quad (2)$$

The derivations of all equations are available in Traynor, et al [27]. Using Eq. 2, the sizes of k and m can be varied according to the needs and constraints of the system designer. After key pre-deployment, nodes in the network discover the keys shared with their neighbors in the *shared-key discovery phase*.

The *session-key establishment phase* attempts to create a secure communication link for pairs of nodes within wireless transmission range that lack a shared key from the previous phase. If it is possible to communicate between a pair of nodes by using a multi-hop path through secure and trusted neighbors, then a key can be generated at the shared neighbor and distributed to the two endpoints. At the completion of this final phase, all nodes within transmission radius of each other should be able to communicate directly to whatever reliability the network designers have specified. Section 4 offers significantly greater details on these two phases.

3.2.1 q -Composite Analysis

Chan et al. [4] extended the basic scheme by requiring nodes to share at least q keys with each other. In so doing, it becomes more difficult for an attacker to compromise communications as q instead of one key must first be captured. While providing robustness to probabilistic keying, the q -composite scheme is not considered for the purpose of node authentication. Specifically, if a node can prove its possession of multiple keys known to be assigned to it, its identity can be probabilistically authenticated. Under the balanced scheme, an increase in q leads to a significant

increase in the number of keys stored by all L1 nodes. However, the unbalanced approach can reduce the burden of the q -Composite scheme on L1 nodes while retaining its security advantages.

The probability that two nodes containing key rings of differing sizes share exactly i keys is:

$$p(i) = \frac{\binom{P}{i} \binom{P-i}{(m-i)+(k-i)} \binom{(m-i)+(k-i)}{m-i}}{\binom{P}{m} \binom{P}{k}} \quad (3)$$

The probability that two nodes share at least q keys with each other is therefore:

$$1 - \sum_{i=0}^{q-1} p(i) \quad (4)$$

We discuss the use of probabilistic authentication in Section 4.

3.3 Key Establishment Trust Models

Because of the assumption of homogeneous network composition, previous keying methods were limited in the ways in which keys could be established. For example, two nodes not sharing a key in the shared-key discovery phase must rely upon a random node with which both source and destination share a key. Whether due to suspicion of compromise, lack of a shared neighbor or resource constraints, such a model of connectivity may not be realistic. We therefore explore three models of key establishment. From these new requirements, we derive equations for connectivity.

3.3.1 Backhaul Trust

In this model, the information collected by L1 nodes is to be backhauled to a resource outside of the local network. In a battlefield setting, this may correspond to a nearby unmanned aerial vehicle (UAV), which relies upon deployed sensors for targeting data. Accordingly, there is no need for an L1 to trust any node but its neighboring L2 node. A key match must there exist between the two. In cases of a single L2 node being present in a neighborhood, we simply apply Eq. 2. If more than one L2 node is present in a neighborhood, two cases exist. In the first, all of the L2 nodes may act as gateways. In the second, only one L2 node will serve as a gateway to the backbone network. In this case, an L1 node may have a key match with the L2 gateway node, or because L2 nodes are trusted, may establish a session key through a path of one or more of the other L2 nodes.

Because the number of keys in the L2 nodes is very high, both theory and practice demonstrate that L2 nodes are able to communicate with each other with a far greater than five-nines reliability;

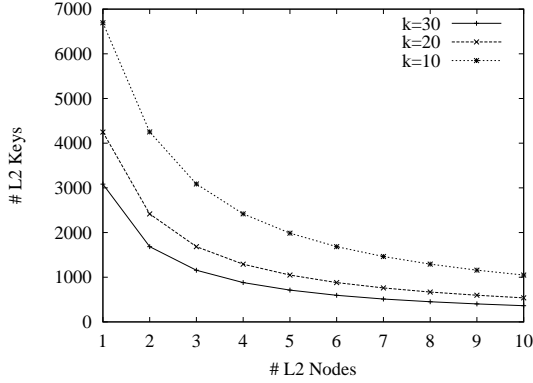


Figure 2: The number of keys in L2 nodes to establish 5-9s connectivity for the backhaul case for varying keys in each L1 node.

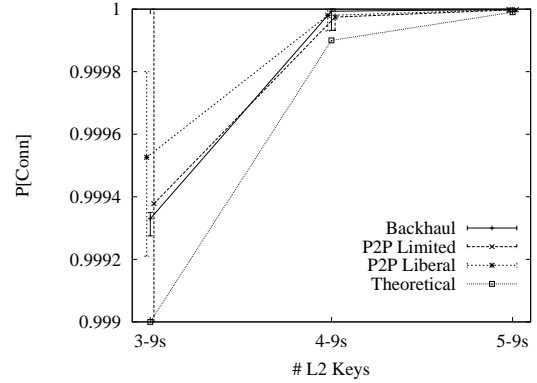


Figure 3: Experimental validation of the calculated key values. Values are staggered to show ranges.

therefore secure links between L2s will always be assumed. In either case, the probability that any L1 node can communicate with the gateway is:

$$P[Conn] = 1 - (1 - U)^g \quad (5)$$

where U is defined by Eq. 2 and g is the number of neighboring L2 nodes. Figure 2 shows the number of keys that must be stored in each L2 node for $k = 10, 20$ and 30 . Figure 3 shows the results of experiments validating these calculated values. All cases were run an order of magnitude more times than the minimum requirement to determine connectivity (10,000 times for 3-9s, etc).

3.3.2 Peer to Peer Limited Trust

In this case, an L1 node desires to communicate with another L1 node in its neighborhood. To do this, it either requires a direct key match with the peer L1 node, or it must be able to establish a session key through a path of one or more L2 nodes. Because L2 nodes are assumed to be more secure due to the presence of key-protecting algorithms, all L2 nodes can be used. The probability of connectivity is therefore:

$$P[Conn] = 1 - (1 - B)(1 - (1 - (1 - U)^g)^2) \quad (6)$$

3.3.3 Peer to Peer Liberal Trust

When using Peer-To-Peer with Liberal Trust to establish connectivity, in addition to using a path of L2 nodes to establish a session key, L1 nodes may establish session keys using a path through a sequence of L1 nodes that provide mutual key matching. Previous studies have considered the

use of up to $n - 1$ hops for the establishment of a session-key. Allowing such a possibly long path to establish session keys has the following two drawbacks. First, the potential latency caused by allowing up to $n - 1$ hops for initializing secure communication may be unacceptably high for the network to complete its mission, especially in cases of high mobility in which nodes may be required to establish new session keys often in the middle of active communication [26]. Second, the introduction of multiple hops may increase the chance that compromised but undetected intermediaries are able to eavesdrop on the actual session-key establishment. While not in the direct path of future packet exchanges, compromised adjacent nodes that assisted with keying may still be able to overhear and decrypt the communications of uncompromised neighbors. Because each hop along an indirect keying path is able to decrypt the data from the previous hop, the probability that at least one compromised node is in a path increases as that path becomes longer.

To estimate the level of connectivity in networks which allow an unlimited number of hops to initialize a session key, we make use of an observation from our simulations. Given a neighborhood of at least 10 nodes and sufficient keys to achieve a relatively high level of connectivity (three-nines), nodes with at least one connection to a neighbor are fully connected to their neighbors. Over the course of 10,000 simulations, we recorded that partitions in the network were limited in size to single nodes. Therefore, for networks with balanced key distributions, Eq. 7 may be used as an approximation for network connectivity when there is no hop limit for establishing keys.

$$P[Conn] = 1 - (1 - B)^{n-1} \quad (7)$$

where B is defined in Eq. 1. This is simply the probability that a node is not isolated.

To estimate the connectivity when limiting the number of hops through which a key may be established to one, we determined the expected number of nodes that any L1 node can communicate with either directly or through one hop. For networks with balanced key distributions, this value is given by Eq. 8.

$$E_{L1}^1 = B * (n - 1) + \sum_{i=1}^{n-1} \binom{n-1}{i} B^i (1 - B)^{n-1-i} E_1 \quad (8)$$

where B is defined by Eq. 1 and $E_1 = (n - 1 - i)(1 - (1 - B)^i)$. For networks with unbalanced key distributions, the value is given by Eq. 9.

Table 1: Number of required keys, 40 node network with 5 L2s

Scenario	Unbalanced		Balanced
	L2 Keys	L1 Keys	L1 Keys
Backhaul	711	30	328
P2P Limited	750	30	328
P2P Liberal, 1-hop	689	30	83
P2P Liberal, n-hop	515	30	54
q-Composite Backhaul ($q = 3$)	2000	30	398

$$\begin{aligned}
 E_{L1}^1 &= E_0 + \sum_{k=1}^g \binom{g}{k} U^k (1-U)^{g-k} (1-B)^{n-1} E_1 + \sum_{k=1}^g \binom{g}{k} \sum_{i=1}^{n-1} \binom{n-1}{i} U^k (1-U)^{g-k} B^i (1-B)^{n-1-i} E_2 \\
 &\quad + \sum_{i=1}^{n-1} \binom{n-1}{i} B^i (1-B)^{n-1-i} (1-U)^g E_3
 \end{aligned} \tag{9}$$

where ,

$$E_0 = B * (n - 1),$$

$$E_1 = (n - 1)(1 - (1 - U)^k),$$

$$E_2 = [(n - 1 - i)(1 - (1 - U)^k)] + [(n - 1 - i)(1 - U)^k(1 - (1 - B)^i)],$$

$$E_3 = (n - 1 - i)(1 - (1 - B)^i).$$

The derivations of Eqs. 8 and 9 are given in Traynor, et al [27].

Given E_{L1} , the expected connectivity of the neighborhood is calculated by Eq. 10.

$$P[Conn] = \frac{E_{L1}^1}{n + g - 1} \tag{10}$$

The memory savings for 5-9's connectivity afforded by the unbalanced approach are summarized in Table 1.

4 Protocol Specification

The specifications for our protocols are now described in detail. For simplicity, the protocol for a network in an infrastructure-less environment will be referred to as LION. The scheme relying upon the presence of the KDC will be referred to as TIGER. LIGER covers the integration of these two components.

The highlights of the protocol operation follow. All nodes are loaded with a random set of keys

drawn from a common pool before being deployed. In addition, the mapping of keys to nodes is stored in a KDC. If the network is operating in stand-alone mode, i.e., with no KDC, we define a protocol to instantiate probabilistic keying. If the network has access to a KDC, we leverage the knowledge of the pre-deployed keys to perform probabilistic authentication with a high degree of confidence. In addition, session keys are established with the enforcement of least privilege. Nodes gather information in this mode of operation so that they may continue to perform some level of probabilistic authentication if the KDC becomes unavailable for periods of time. The mode of operation may change between stand-alone and KDC-mode.

The portrait of sensor networks painted by most of the current literature is one of extremes. Systems exist either in total separation from infrastructure and intervention or with constant access to such resources. Networks designed to operate in isolation therefore never consider harnessing new resources as they become available. Likewise, systems designed with a reliance upon available infrastructure flounder in its absence. In reality, large-scale sensor networks will have to optimally perform their missions in both of the above settings. If, for example, there is a method of transmitting data from a sink node to some external destination, then the ability of a sensor node to communicate with a KDC is entirely realistic. Indeed, if data cannot be drawn out of a sensor network and delivered to some distant location, the usefulness of the network is extremely limited.

Simply placing either only LION, TIGER or any other single-mode scheme in a sensor node therefore fails to fully utilize the potential of these networks. The answer, however, is not using combined, unaltered versions of both LION and TIGER in each sensor. Such a solution fails to take advantage of redundant or similar mechanisms. Specifically, including a unique master key for the purpose of authentication via a KDC fails to take into account the potential size of the code supporting this an additional protocol. Because the highly constrained memory of L1 nodes is one of the chief concerns of all solutions implemented on these platforms and an equally effective solution can be achieved probabilistically, a hybrid method of key management becomes the most efficient solution for such a setting.

The combination of slightly modified versions of these two schemes results in LIGER - a more

robust method of key management for heterogeneous sensor networks. The combination enables different levels of probabilistic authentication without increasing memory requirements of the L1 sensor nodes.

We first describe the stand-alone component of the system (LION), followed by the KDC-based component of the system (TIGER). We then describe probabilistic authentication and its application to LIGER.

Notation

- A, B are principles (e.g. communicating L1 nodes)
- $ID_{A_0}, \dots, ID_{A_{k-1}}$ are the sorted key identifiers corresponding to the keys held by node A .
- K_A is a secret key known by node A .
- $K_{A,B}$ is a session key shared between nodes A and B .
- K_{A_AUTH} is an authenticator key for node A .
- K_{A_i} is some key corresponding to an ID from within the range described directly above.
- $L1$ is a sensor node.
- $L2(GW)$ is the L2/Gateway node.
- $MAC(K_A, R|S)$ is a Message Authentication Code of the values R and S , using key K_A
- MAP_A is the bitmap corresponding to a sorted representation of $ID_{A_0}, \dots, ID_{A_{k-1}}$.
- N is a nonce.
- $\{S\}_{\langle K \rangle}$ is a value S encrypted in key K .

4.1 LION: Standalone Key Management

After deployment under the LION protocol, an L1 node learns its neighbors through the exchange of `Hello` messages², and then attempts to establish keys with them. To accomplish this, the node broadcasts all of its key identifiers. Because the keys themselves are not transmitted and similar information could be gathered from a traffic analysis attack [9], this method does not compromise the integrity of the node itself. If a neighboring node overhears this transmission and determines

²This could also be accomplished by passive means.

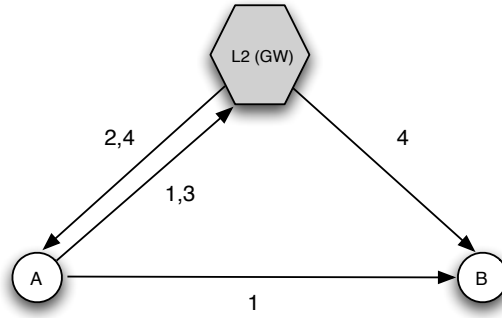


Figure 4: LION key establishment.. First, L1 node *A* broadcasts out its key identifiers. A neighboring L2 determines it has a match with the L1 and sends a challenge/response message.

that it shares one of the keys associated with the broadcast, it responds to the source with a challenge/response. Other methods of determining the keys of neighbors including the use of hash functions [33], encrypted broadcasts [9, 4], and the use of polynomials [2].

Messages 1 and 2 in Figure 4 show the message flow for the case in which a node, *A*, has a key match K_{A_i} , with a L2 node. The messages exchanged between the two exhibit the following format:

- 1) $A \rightarrow * : A, N, ID_{A_0}, \dots, ID_{A_{k-1}}$
- 2) $L2 \rightarrow A : A, L2, N, ID_{A_i}, \{ID_{A_i}, L2, N\}_{\langle K_{A_i} \rangle}$

L1 nodes amass a list of neighbors with which they do and do not share keys. When the shared-key discovery phase ends, a node attempts to use the neighbors with which keys are already shared to assist it in establishing secure connections with all neighbors. In the Limited Trust model discussed here, this “Request for Assistance” (which contains all of the node IDs with which a secure relationship has not been established) is sent directly to an L2 node. The L2 node, having already established a link with the targeted L1, transmits a message to the requester and targeted node containing a session key encrypted in each of the keys shared with the L2 node. Each L1 node then receives the L2 broadcast, decrypts the session key and begins the secure transmission of data. The messages for the indirect phase are:

- 3) $A \rightarrow L2 : A, B, N$
- 4) $L2 \rightarrow * : A, B, N, \{K_{A,B}, N\}_{\langle K_{A,L2} \rangle}, \{K_{A,B}, N\}_{\langle K_{B,L2} \rangle}$

Messages 3 and 4 in Figure 4 show the indirect phase of the protocol with the “Request for Assistance” message transmitted to the neighboring L2 node following the Limited Trust model.

This message would be broadcast to all neighbors if a less stringent trust model was in effect.

If a node assists in establishing a session key during the indirect phase of the protocol, it deletes this key as soon as end-to-end communication is established. The two endpoints of communication also re-key immediately. In this way, if a node is compromised, it will not contain any valid session keys other than its own. A more detailed discussion of compromise is given in Section 6.

4.2 TIGER: KDC-Based Key Management

In locations such as “smart buildings” or factories where sensor networks may be used to gather data corresponding to changing environmental, structural, and inventory-related conditions, access to a KDC is an entirely realistic assumption. We have designed TIGER for this scenario. Because this protocol is similar to most KDC-based systems we provide only a brief description here.

Before the system is initialized, each L1 node is bootstrapped with the same set of k keys as with LION. L2 nodes share a public/private key combination with the KDC.

To perform authentication, each node creates an authenticator key from a combination of their pre-deployed keys as described below. After discussing the basic authentication mechanism, we give a detailed definition of this protocol.

4.2.1 Probabilistic Authentication

One of the chief goals in the development of sensor network security is the minimization of memory overhead. Specifically, if the ability of an L1 node to perform its sensing task is limited by the memory footprint of a security solution, the security solution should be considered ineffective. Because one of the primary occupiers of memory in random pre-distribution schemes is the keys themselves, all efforts must be made to decrease this burden on the platform. Accordingly, LIGER only stores a single set of keys for use in both the LION and TIGER portions of operation. To provide robust operation in the face of disconnection with the KDC, these keys are pre-deployed as in the LION method.

In order to prove its authenticity, a node instantiates a temporary *authenticator key* by which the system may perform probabilistic authentication. This key is created using a simple operation on a subset of the k pre-deployed keys in each L1 node. This scheme is also used for L1 nodes to loosely authenticate each other via an L2 node when a KDC is unavailable.

In TIGER, each L1 node uses q of its k pre-deployed keys to generate the authenticator key. The key itself is created by performing an XOR on the selected q keys. This operation is guaranteed to create an unguessable, pseudo-random value from the key space as demonstrated by Shannon [25]. Because an attacker must know all of the key values associated with the creation of an authenticator key in order to derive it, this system is protected to a threshold of $q - 1$ for any given node. The hardness of breaking an authenticator key, for some $q < k$, is further enhanced as discussed in the protocol definition by allowing the subset of keys from which the authenticator key is derived to be changed as described below in the protocol specification. An analysis of the robustness of the authenticator key is given in Traynor, et al [28]. Because the KDC knows all k keys pre-deployed in each sensor node, it can compute authenticator keys, and thus authenticate each L1 node in the network.

One drawback of the original q -composite method is that it requires an increase in the number of keys deployed in a L1 node to provide a reasonable probability of obtaining q key matches. When using a KDC in our system, no additional keys are required in the L1 nodes to maintain the likelihood of q key matches between a L1 nodes and a KDC because the KDC knows all of the keys deployed in the L1 nodes. In addition, because with unbalanced key distribution only a small number of keys are deployed in L1 nodes, the likelihood of one L1 node having q keys in common with a second L1 node and thus being able to impersonate it, is infinitesimally small.

4.2.2 TIGER Protocol Definition

TIGER takes advantage of a KDC with a protocol enforcing least privilege over key establishments while retaining the ability to operate should the connection to infrastructure cease to exist. We discuss the case in which nodes are activated and a link to the KDC is available through an L2 node. If conditions prevent a connection being established, as is the case in the military deployment example, the network defaults to the LION protocol until a KDC link becomes available.

L1 - L1 Authentication and Key Establishment. An L1 node A wishing to establish a secure and authenticated session key with a neighboring L1, a node advertising itself as B , begins the process by creating a token. The token itself is the MAC of a series of values included in the initial packet - the principles involved in the exchange, a nonce, and a sorted bitmap of the keys used to create

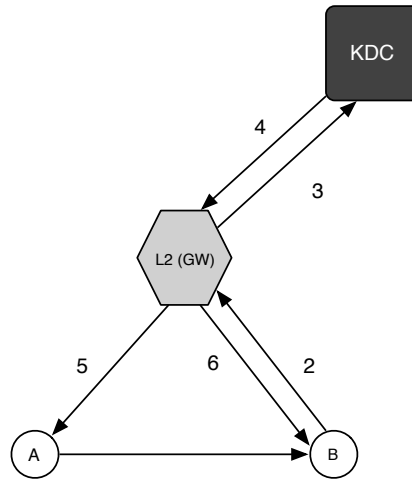


Figure 5: The TIGER message flow for key establishment between L1 nodes. Node *A* sends a token to Node *B*. *B* includes its own token and forwards that message to the KDC. The KDC validates both tokens and returns a copy of the a session key encoded in one of the allowable authenticator keys to both *A* and *B*.

the current authenticator key. Upon receiving the token, the node believed by *A* to be *B* makes a decision as to whether or not it desires an authenticated connection with the node it believes to be *A*. For example, if *B* has low battery power, is already congested with large amounts of data from other neighbors or has judged node *A* to be compromised [17, 30], it may not wish to establish a key with *A* and thus drops the request.

If *B* decides to set up an authenticated relationship with *A*, it includes the token sent by *A* with its own token in a message to an L2 node. The L2 node then forwards the packet on to the KDC. The validity of the two tokens is determined by generating the appropriate authenticator keys for both *A* and *B* according to the sorted bitmaps of the identifiers corresponding to keys used to make each token. If both tokens are deemed legitimate, the KDC responds with a message to the L2 containing a copy of a session key encrypted in a new, randomly chosen authenticator key for both *A* and *B*. The message from the KDC will also contain a bitmap corresponding to each of the authenticator keys used to sign the session keys. Nodes *A* and *B* then receive a transmission from the L2 node, generate the appropriate authenticator keys to unlock the session key and begin communication. The messages for this protocol flow as shown in Figure 5 and appear as follows.

- 1) $A \rightarrow B : A, B, N, MAP_A, MAC(K_{A_AUTH}, A|B|N|MAP_A)$
- 2) $B \rightarrow L2 : A, B, N, MAP_B, MAC(K_{B_AUTH}, A|B|N|MAP_B), MAP_A, MAC(K_{A_AUTH}, A|B|N|MAP_A)$
- 3) $L2 \rightarrow KDC : \text{Forward Message 2 to KDC}$
- 4) $KDC \rightarrow L2 : A, B, N, MAP_{A'}, \{K_{A,B}, N\}_{(K_{A'_AUTH})},$
 $MAP_{B'}, \{K_{A,B}, N\}_{(K_{B'_AUTH})}, MAC(K_{A,B}, A|B|N|K_{A,B})$
- 5) $L2 \rightarrow A : A, B, N, MAP_{A'}, \{K_{A,B}, N\}_{(K_{A'_AUTH})}, MAC(K_{A,B}, A|B|N|K_{A,B})$
- 6) $L2 \rightarrow B : A, B, N, MAP_{B'}, \{K_{A,B}, N\}_{(K_{B'_AUTH})}, MAC(K_{A,B}, A|B|N|K_{A,B})$

Requiring the KDC to create a new authenticator key allows the protocol to supplementary harden the system against an attacker compromising multiple nodes in attempt to forge an identity. For example, a valid authenticator key could be generated by the KDC from any of the elements available in the compliment of the bitmap of the original message. Furthermore, it allows for key revocation protocols to exclude the use of keys specifically known to be compromised. Such a policy would not need to be enacted for the generation of authenticated session keys until all keys associated with the authenticator key are compromised. This will extend the lifetime of a network.

This scheme is similar to Kerberos in that it requires a ticket from a node by which it may be authenticated before a session key is granted. In TIGER, however, both nodes in which a session key is being established are required to provide a token to the KDC. We feel this protocol is more suited to a peer-to-peer environment, and prevents a single node from easily generating a large amount of requests to a KDC to receive session keys to nodes that are not interested in communication. Like Kerberos, TIGER requires two messages from the clients to establish session keys; however, in TIGER each peer generates one message as opposed to Kerberos in which a single client generates both messages. TIGER therefore balances message load and energy consumption across the network more efficiently.

L2 - L1 Authentication and Key Establishment. An L2 node wishing to view data collected by an L1 node must broadcast `Hello` messages in order to alert the L1 nodes of its presence. The L1 node A provides the L2 node with a token/MAC created in the same way as described above;

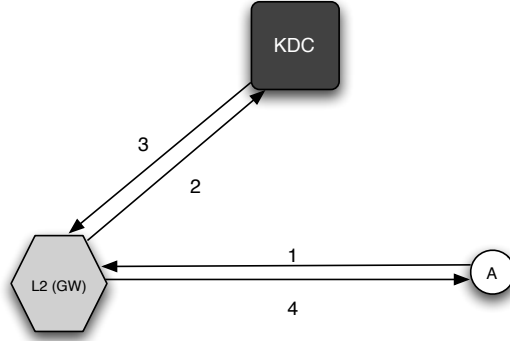


Figure 6: The TIGER message flow for establishing keys between L1 and L2 nodes. By requiring the L2 node to include a token generated by an L1, least privilege is enforced.

the L2 node forwards the contents of this message on to the KDC. If the KDC is able to verify the MAC, the L2 node will have verified that it is indeed in contact with node A .

The KDC then returns a message to the L2 node containing a session key and a copy of the session key encrypted with the authenticator key of A . This information is included in a response to A , which also contains a MAC of the packet contents calculated with the KDC-generated session key. The last MAC can be confirmed as having been created by the L2 node after A has decrypted the session key. Because the L2 node can not establish keys with the other remaining nodes in the network without their direct participation, least privilege is preserved. The messages to implement this protocol follow the flow shown in Figure 6 and use the format below:

- 1) $A \rightarrow L2 : A, L2, N, MAP_A, MAC(K_{A_AUTH}, L2|N|MAP_A)$
- 2) $L2 \rightarrow KDC : \text{Forward Message 1 to KDC}$
- 3) $KDC \rightarrow L2 : A, N, MAP_{A'}\{K_{A,L2}, N, \{K_{A,L2}, N\}_{\langle K_{A_AUTH} \rangle}\}_{\langle K_{KDC,L2} \rangle}$
- 4) $L2 \rightarrow A : A, N, MAP_{A'}\{K_{A,L2}, N\}_{\langle K_{A'_AUTH} \rangle}, MAC(K_{A,L2}, A|N|MAP_{A'}, \{K_{A,L2}, N\}_{\langle K_{A'_AUTH} \rangle})$

5 Performance Evaluation

We now present the experimental results of the LIGER system implementation. In the first subsection we present benchmark results for processing on individual nodes. In the next two subsections we present results on network initialization times obtained via simulation using TOSSIM [14].

Table 2: Microbenchmark results for LION and TIGER modes.

LION			
Operation	Mica2 (μsec)		Stargate (μsec)
RC5 Key Setup	5720.7	($\sigma=21.2$)	24.7 ($\sigma=1.3$)
MAC Initialization	11060.0	($\sigma=6.6$)	24.6 ($\sigma=1.1$)
MAC (54 Byte Input)	15953.3	($\sigma=29.9$)	42.6 ($\sigma=1.2$)
RC5 Decryption	3312.5	($\sigma=12.6$)	6.4 ($\sigma=0.9$)

TIGER			
Operation	Desktop KDC (μsec)	Desktop GW (μsec)	Stargate GW (μsec)
Public Key Encrypt	169.6 ($\sigma=8.9$)	124.5 ($\sigma=64.0$)	1823.4 ($\sigma=18.9$)
Private Key Decrypt	4065.4 ($\sigma=66.2$)	4310.4 ($\sigma=748.2$)	76164.2 ($\sigma=15891.2$)
Execution Time	4588.1 ($\sigma=107.9$)	32450.4 ($\sigma=25080.8$)	112370.7 ($\sigma=30478.5$)
RC5 Encryption	3.6 ($\sigma=0.5$)	3.6 ($\sigma=0.5$)	6.6 ($\sigma=0.9$)

5.1 Node Benchmarks

In this section, we discuss the implementation of the LION, TIGER and LIGER components for our hybrid key management system. The Crossbow Mica2 mote [5], with a 4 MHz Atmel ATmega128L processor, 128 KB of program Flash memory, 512 KB of measurement flash memory, and a 916 MHz ChipCon radio is used as the platform for L1 nodes. The L2 node consists of a Mica2 mote mated with a Crossbow Stargate with a 400 MHz Intel PXA255 Xscale processor, 64 MB of SDRAM and 32 MB of flash memory. Additional tests were run using an L2 node with the Stargate replaced with a desktop computer with a 2.80 GHz Intel Pentium 4 processor and 512 MB of RAM running Fedora Core 2 with Linux Kernel 2.6. Where applicable, the KDC was executed on a desktop computer with the same configuration as described above.

A series of microbenchmarks were conducted in order to characterize the load placed upon each of the platforms. The average of these timing experiments, which were recorded over 10,000 iterations of the protocols (σ = standard deviation), are shown in Table 2.

The timing comparison between the two potential gateway platforms for TIGER illustrates the tradeoffs between performance, portability and expense. For example, the processing time of the L2 gateway function on the desktop is more than three times faster than the Stargate version. While a laptop computer could certainly be equipped with the same specifications as this desktop, placing such a device in an unattended setting may not be a realistic option. Furthermore, it may or may

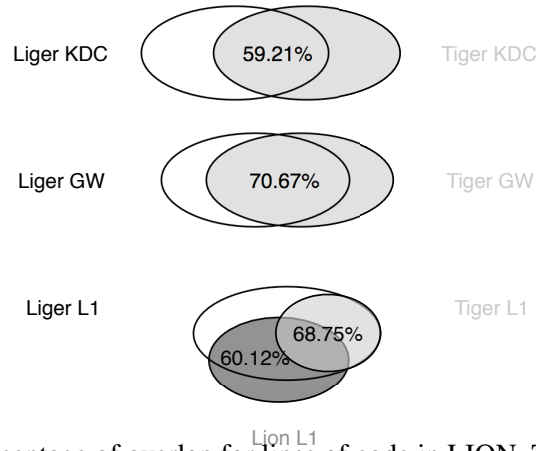


Figure 7: The percentage of overlap for lines of code in LION, TIGER and LIGER.

Table 3: L1 code size on Mica2 motes

Scheme	Size in ROM	Size in RAM
LION	19636	522
TIGER	16148	458
LIGER	20982	532

not be desirable to provide a user interface as part of the gateway device. The implications of particular platforms should therefore be carefully considered before deployment of each system.

A second observation of TIGER is that the processing associated with the secure link between the L2 node and the KDC accounts for approximately 70% of all processing on the Stargate gateways. An obvious improvement is to use a symmetric key between the KDC and GW. This reduction would allow the Stargate to process packets at rate greater than their arrival, thereby making it equivalent to the desktop option. If the Stargate were the cheaper of the two platforms, the network could then be constructed for a reduced cost without negative consequences to performance.

From the results of the LION benchmarking it is evident that the L1 nodes are a processing bottleneck. This further supports the unbalanced key distribution design of LION in which L2 nodes offload a great deal of processing from the L1 nodes.

Lastly, we consider the memory footprint of implementing a multi-modal protocol. Table 3 offers a view of the code sizes. Through the careful reuse of similar functions, as shown in Figure 7, the combination of both LION and TIGER protocols requires only slightly more storage space than independent implementations of each.

5.2 Network Initialization Results

5.2.1 Simulation Model

We focus on the initialization time using LION as it places the highest processing burden on the network and nodes. Also, it is likely in many scenarios in which a network deployment is required in response to an emergency, that the network may be deployed in an ad hoc fashion without initial access to a KDC. We specifically compare the performance of the balanced and unbalanced key distribution strategies.

As the TinyOS packet size is limited to 29 bytes of data and L1 nodes may contain between 10 and 328 keys, multiple packets must be broadcast to advertise all identifiers. Because a continuous sending of these packets fails in TOSSIM (and since spin locks do not exit in TOSSIM), we employed a daemon mechanism to handle sending packets from a node. A node maintains a list of packets to be transmitted. Whenever a packet to be sent is generated by the node, it is appended to this list. The packet at the head of this list is delivered to the medium every α seconds. We set $\alpha = 20 \text{ msec}$ in our simulations because this provides sufficient spacing for TOSSIM while still allowing for maximum channel utilization.

We pre-deploy a sufficient number of keys in each node to provide 0.99999 network connectivity as described in Section 3.3. A network is considered to be initialized when all nodes have established keys with at least 90% of their neighbors. In all scenarios tested, we fix the total number of nodes to 100, each with a transmission range of 50 feet. In order to simplify simulations, all nodes in the network were assumed to have the same processing power. As real Stargate nodes will achieve much lower processing delays than the Mica2 motes, our results are very conservative.

We use “passive” key establishment to further improve performance. Suppose A broadcasts its key IDs. B replies to A , which gets an active key match with B . At the same time, B gets a passive key match with A since B knows which keys A has and then can find a match key. This will reduce the communication overhead and hence reduce the network initialization time. For the active key match, A challenges B , so it is sure about this key match. For the passive key match, B did not challenge A , so it is not sure about this key match since someone else can launch an attack

by using A 's ID. As a result, this passive key match may not be a real match. However, this is not necessarily a problem since B can challenge A during its direct phase. Additionally, the "Request for Assistance" message could be encrypted in the key shared between the two parties, A is able to demonstrate its knowledge of the shared key with B . If we assume no compromise during network initialization, as is often done in this area [31, 32], passive matching is a good solution.

5.2.2 Parameter Setting

One immediate observation from the simulations was that due to the nature of the key establishment protocols, many collisions occurred on the air interface during network initialization. To limit the number of collisions, nodes broadcast key requests with an initial random jitter. We chose the jitter value based on simple analysis and experimentation. Given the data rate of the wireless interface and the number of packets broadcast per node, with perfect scheduling it requires approximately 40 seconds for all nodes to complete their broadcasts. We ran simulations for jitter times of 30-60 seconds. With a random jitter of 40 seconds or lower, we found that the number of collisions that occurred precluded nodes from reaching their expected level of key matching in each round, thus delaying network initialization. For values of greater than 50 seconds, we found little improvement in connectivity over the case of a 50 second jitter. Therefore, to keep the total delay of each phase low, and to allow nodes to reach their ultimate connectivity quickly, we set the jitter to 50 seconds.

The necessity for this additional jitter highlights the need for more resilient MAC layer protocols in sensor networks. Because events, in this case the establishment of keys, are likely to create a significant amount of traffic, it is critical that each layer is optimally designed to maximize the use of the spectrum. From the results of this work, it becomes obvious that a backoff algorithm that spaces retransmission attempts out more evenly would be extremely valuable.

5.2.3 Results: Initialization Time

In order to determine the amount of time required for the direct and indirect phases of LION, we set inter-phase timers to large values such that each stage becomes easily discernible. Figure 8 demonstrates the separation of the direct and subsequent indirect phases on networks implementing the balanced and unbalanced keying schemes with 5 L2 gateway nodes deployed. Each point on the

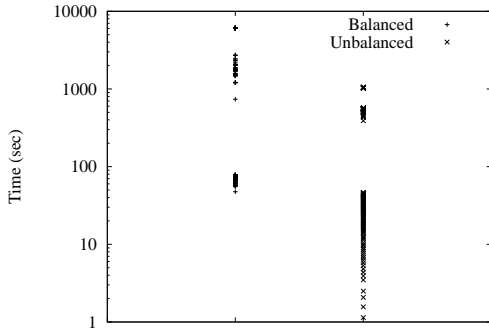


Figure 8: The termination of individual nodes in the network for both the direct and indirect phases of initialization. Subsequent phases are spaced at 400 seconds.

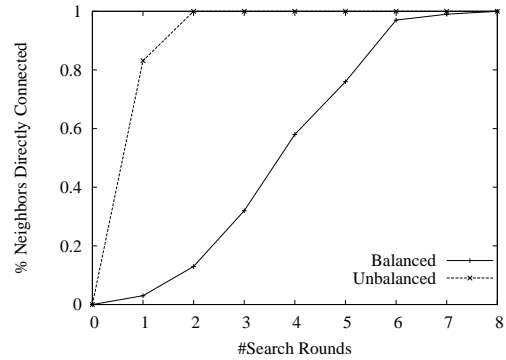


Figure 9: The percentage of nodes with 90% connectivity directly to their neighbors across a number of key establishment rounds.

graph denotes a node completing the phase by achieving at least 90% connectivity (key matching) with its neighbors. The cumulative distribution function of the network initialization points is shown in Figure 9.

As seen in Figure 8, the balanced scheme requires eight phases (direct and seven indirect) for all nodes to achieve the target connectivity, while the unbalanced case requires three phases. The additional phases in the balanced case were required for two reasons. First, because nodes only have a 0.5 probability of having a key match directly with a neighbor in this case, multiple rounds of the indirect phase of the protocol are required for nodes to assist in establishing keys. Second, due to the large volume of traffic, many collisions occur despite the random jitter, further reducing the probability of successfully finding key matches in each round. In the unbalanced case, nodes have a high probability of having a key match with an L2 gateway, so fewer rounds are required.

Efficiently setting timers to achieve minimal inter-phase timeouts in real networks is challenging. Short timers increases the number of nodes competing for the medium and therefore adds the potential for more collisions and necessitates a potentially significant number of retransmissions. Setting timers too conservatively unnecessarily increases the time required to bootstrap the network. In a manner reminiscent of setting the retransmission timers for TCP, we set the interphase timer to be the sum of the average stage termination time of each phase and a multiple of the stan-

Table 4: Phase timing data (5 ft spacing)

Scheme	Init (sec)	Init + 0.5σ	Init + σ	Init + 4σ
Balanced	1865.17	1887.394	2097.507	3358.183
Unbalanced	274.667	313.846	331.800	439.182

Table 5: The effects of varying the number of L2 nodes.

# L2 Nodes	Initialization Delay (sec)	\bar{x} Messages/L1
0(Balanced)	1865.170	1199.180
1	355.038	62.141
2	336.178	61.704
5	274.667	58.968
10	244.910	56.956

standard deviation (σ) of the termination time. Based on several experiments reflected in Table 4, we choose the average time plus 0.5σ as the timer value.

Table 5 demonstrates the average initialization times for grid networks containing 0, 1, 2, 5 and 10 L2 nodes with inter-node spacing at five feet. The case of 0 L2 nodes corresponds to a balanced key distribution. The relationship between initialization time and the number of L2 nodes is inversely proportional; however, the addition of L2 nodes to the network represents an increase in the cost of deployment. In order to balance robustness to failure with economics, the remainder of the experiments involving an unbalanced key distribution therefore assume the presence of five L2 nodes per neighborhood.

Figure 10 shows the network initialization times for varying node densities for both the balanced and unbalanced (5 L2 gateways) cases using both times based on perfect knowledge and via the method described in Table 4. As is evident, the unbalanced key distribution provides much lower network initialization times for dense sensor networks. The main reasons for this are the reduced number of rounds required when using the unbalanced key distribution as shown in Figure 8, and a reduced number of packet collisions as discussed below.

While our network initialization time results are shown for cases in which 90% key matching with neighbors is required in a network, random graph theory [9] tells us that a node only needs to be directly connected to a small subset of its neighbors for a network to be fully connected. Establishing so few direct links in a real network decreases the opportunities for optimal routing. Our

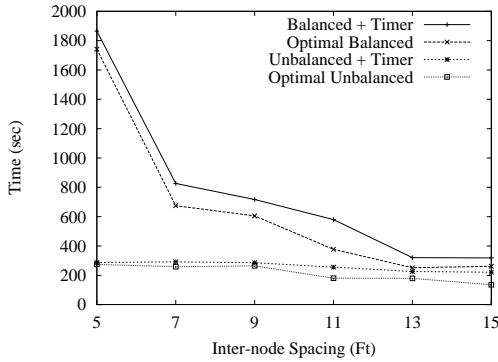


Figure 10: Balanced and Unbalanced network initialization times for the LION scheme.

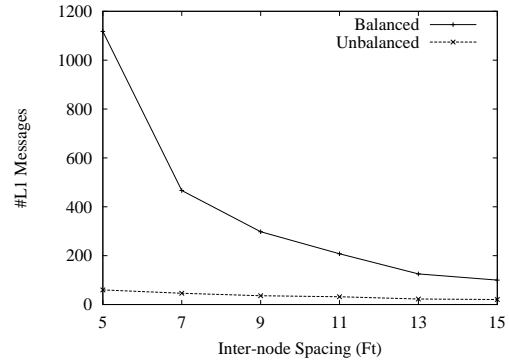


Figure 11: The average number of messages per L1 node to initialize networks.

experiments therefore strive to achieve 90% direct connectivity between nodes and their neighbors in order to minimize path lengths. Such initially high, direct connectivity may be unnecessary in many networks. For example, the administrator of a network may deem that having all nodes establish secure relationships with at least 60% of their neighbors is enough to meet some expectation of performance. Alternatively, a network could be deployed in an emergency situation and would therefore strive to gain the maximum secure connectivity possible within a given time limit.

5.2.4 Results: Message Complexity

Figure 11 shows the number of messages that must be broadcast by the L1 nodes in order to achieve secure relationships with their neighbors. In the worst case, each L1 node in the balanced case is required to send two orders of magnitude more messages than in the unbalanced case. As transmission bandwidth is limited (38.4 kbps theoretical maximum), the sheer number of packets needed to establish keys is overwhelmingly the source of delay in the system. This problem is not realistically solved by the introduction of higher bandwidth radios such as those included with the new MICAz [5]. Due to the power constraints inherent to wireless sensor devices, the number of packets that must be transmitted is far too expensive for real implementations. The balanced key management approach is therefore inappropriate for most dense sensor networks.

Performance improvements in these networks, however, are not only limited to decreased packet volumes. The nature of the unbalanced key management system implemented in LION

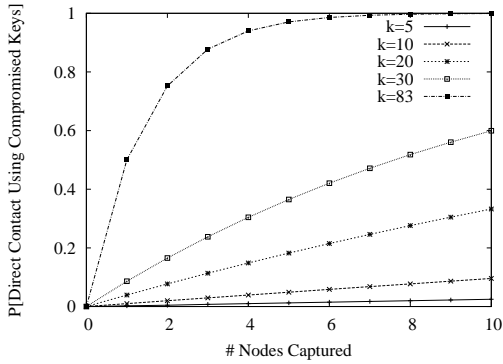


Figure 12: The ability of an attacker to establish new keys with their neighbors given multiple compromised L1 nodes.

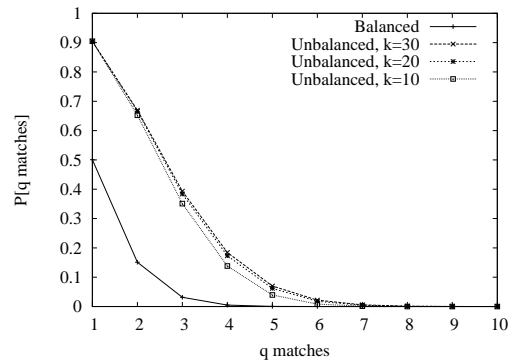


Figure 13: The probability that two nodes can match q keys with each other for the balanced (83 keys/node) and unbalanced (30 keys/L1; 750 keys/L2) cases.

is such that certain nodes throughout the network, specifically L2s, are expected to process an elevated level of packets compared to their neighbors. Table 5 demonstrates decreased network quiescence time without a significant decrease in the number of messages transmitted by L1 nodes. This reduction is directly proportional to the number of L2 nodes sharing the processing load. The addition of L2 nodes to real sensor networks would therefore have performance benefits additional to those recorded via TOSSIM.

6 Discussion

6.1 Node Compromise

Recall that all communicating nodes establish a new key per session, even if they have a direct key match. Further, all nodes that assist in establishing session keys during the indirect key matching phase delete the session keys they assign once key initialization is complete; in addition, the communicating end points re-key immediately. Therefore, compromised nodes cannot eavesdrop on any ongoing communication other than that which is directly terminated on the compromised node. The greater danger is that these compromised nodes may establish new session keys to inject false data into the network, or assist in establishing new session keys for neighbors and be able to eavesdrop on new sessions. We begin by analyzing the ability of compromised L1 nodes to communicate with their neighbors. The results are summarized in Figure 12.

Superficially, it would appear as if an attacker need only compromise a small number of L1

nodes in order to detrimentally affect the network. The modifications to the network model, however, offer additional protections to the network. Intrusion detection schemes successful in ad hoc networks [17] can now be applied to protect sensor networks. Additional help can potentially be offered by infrastructure-based resources now available through the backhaul link.

Even in the absence of compromise detection, authentication via the q -composite scheme protects network integrity from a large number of compromised nodes. As is shown in Figure 13, nodes with an unbalanced distribution of keys can probabilistically authenticate each other in the absence of a KDC with much higher probability than nodes with balanced distributions. Attempting to forge identity or eavesdrop by compromising specific keys is also difficult. For example, in a network of 1,000 nodes (12.5% being L2s), each key is likely to be located in approximately 1% of the nodes on average. An attacker selecting nodes randomly would therefore have to physically compromise almost 100 nodes (12.5% being L2s) in order to locate a specific key. Searching for q specific keys in order to impersonate a particular identity is therefore extremely costly to an adversary. Moreover, if an attacker is able to compromise nearly 100 nodes in a network without being detected, the system is likely facing far more critical problems. The drastic reduction in the cost of the q -composite scheme due to network heterogeneity therefore further bolsters the strength of this probabilistic keying method.

The compromise of an L2 node is, of course, potentially profoundly damaging to the system. These nodes are responsible for the majority of keying material and potentially connectivity with other areas of the network. The presence of such nodes, however, presents more opportunities for the advancement of security in this setting than are obvious at first glance. For example, under the homogeneous model, the goals of tamper resistance and cost minimization are in direct competition. Under the heterogeneous model, only a small portion of the nodes in the network would require such protection. The total cost of effectively providing tamper Resistance to the network is therefore greatly reduced. In concert with all of the previously mentioned benefits, the overall security of a system built on the principle of heterogeneity is significantly greater than its homogeneous counterpart.

6.2 Switching Modes of Operation

The advantage of LIGER is that it allows a sensor network to operate in a secure and efficient manner regardless of the available resources. There are, however, a number of tradeoffs experienced by a system operating in either mode. A comparison of these issues is made below so as to further clarify the effectiveness of the mechanisms provided by both LION and TIGER. Specifically, we examine the effects of transitioning between modes and discuss how security is affected.

TIGER to LION: An example system likely to initialize using TIGER and transition into the LION protocol is a sensor network that is deployed in support of a planned operation. In this case, session keys may be initialized in a controlled environment with access to a KDC. As the operation progresses, it is possible that access to the KDC is lost.

In the ideal setting, a sensor network is allowed to initialize in the presence of KDC. Every node in the network is able to authenticate each of its neighbors to the full extent supported by this system. Because the KDC knows all of the keys stored in both the L1 and L2 nodes, it can send the key identifiers common to an L1/L2 pair to an L2 node in message 4 of the TIGER protocol flow. If the system later transitions into the LION protocol, either by design or out of necessity, the stored, authenticated key identifiers now in the L2 node can be used for performing authentication of L1 nodes when refreshing expired keys or helping to establish an authenticated connection between two L1s without the presence of a KDC. The limitations and benefits of this approach are discussed in detail below.

If the KDC is cut-off, the L2 node will have a list of the i matching keys it has with each L1 node. It can use these i keys to challenge the L1 nodes in order to authenticate them. The main limitation of this mode of operation is that in many cases the value of i will be small.

As shown in Figure 13, if 750 keys are deployed in an L2 nodes and 30 keys are deployed in an L1 node, the probability of the L2 having two matching keys with a particular L1 node is 67%; the probability of three keys matching is 39%. While these values show that in many cases a L2 node will not be able to authenticate a L1 node by challenging with multiple keys, they are still high enough that if a node does have two or three key matches with the L2 node, the L2 node

can be highly confident that the node is not being impersonated. With these key deployments, the probability of two L1 nodes having the same two or three key matches with a L2 node is $8.70 \cdot 10^{-6}$ and $2.44 \cdot 10^{-8}$, respectively.

If, on the other hand, the L2 node is deployed with a much larger number of keys, the number of keys it has in common with L1 nodes may be much higher. In this case, the L2 node may decide to cease using the KDC by design. The benefit of this approach would be that the authentication would be performed locally and the network delay incurred by accessing a KDC would be removed.

A benefit of using the initial KDC connectivity to inform L2 nodes of the keys deployed in L1 nodes is that L1 nodes will no longer be required to broadcast their key IDs in stand-alone operation to establish session keys with L2 nodes. From a security perspective, this reduces information leakage from the system. The details of this leakage are discussed in the reverse transition below.

LION to TIGER: A network operating in the military scenario suggested in the Introduction would likely begin secure operation via the LION protocol. Because of the lack of friendly troops with connections to a backbone network and the need for a rapid deployment, initial access to a KDC may not be possible.

The difficulty with the LION scheme, while providing security in the absence of a KDC, is that its ability to truly authenticate nodes is more limited. However, some level of authentication is possible if we assume, like a number of other schemes [32], that the initial broadcast of key identifiers is conducted during a network bootstrapping phase wherein all nodes are free from compromise. During this bootstrapping phase, nodes can create a list of key identifiers present in each node. After that period, authentication of new connections can be achieved via the comparison of keys known to be shared with a neighbor versus those used to sign or encrypt a message. As would be expected, the likelihood that two L1 nodes have a large number of key matches to create relatively strong authenticator keys is unlikely. However, if unbalanced key distribution is used, L2 nodes will have a reasonable probability of having multiple key matches with L1 nodes.

The main drawbacks of this scheme are the reliance on the assumption that nodes are not compromised during the start up period and that the key identifiers must be broadcast. As mentioned

earlier, broadcasting key identifiers does not explicitly reveal any information about the value of keys that cannot be gained from a traffic analysis attack [9]; however, information is still being leaked to an adversary through this approach. If an attacker is able to compromise a large number of nodes, it becomes possible to use the key identifiers to pick a node to impersonate. Schemes where key identifiers are assigned via a hash function [33] exacerbate this problem further by allowing an attacker to determine which nodes they can impersonate without ever having been near that target. While many have argued against the use of a broadcast mechanism for the distribution of key identifiers, it forces an adversary to have physical proximity to a node it intends to impersonate, thereby making the system more robust.

Because it is easier to impersonate another node while a network is running in LION mode, an administrator may consider forcing all nodes in the network to re-establish keys if a KDC becomes available. Assuming that all k keys within an L1 node were not known to the adversary, the system prevent malicious nodes added by the attacker from injecting further data into the network. This purging of the system, of course, comes at the cost of the additional overhead associated with re-initializing an entire network.

In summary, it is always advantageous to initialize a network using TIGER. In this mode nodes may be authenticated with a confidence level as an attacker was required to guess a 128 bit key. Also information may be securely distributed to L2 nodes so that some level of authentication may be performed if the KDC becomes unavailable. If the network must initialize using LION, some level of authentication may still be performed, but some information may be leaked to adversaries.

7 Conclusion

In this paper, we have leveraged the emerging trend of heterogeneity in sensor networks to provide new, more efficient mechanisms for secure communications. We began by introducing the probabilistic unbalanced key distribution. In this scheme, nodes with more intrinsic resources are responsible for a greater proportion of the communications and memory overhead associated with security. Second, we proposed a number of trust models for key establishment. The combination

of these two techniques allows for the administrator of a system to implement policy at a much finer granularity and lower cost than in previous systems. Third, we designed a multi-modal key establishment protocol. Whether in the presence or absence of a backhaul link, nodes running the LIGER suite are able to take advantage of all available security resources in potentially dynamic environments. Fourth, we performed extensive evaluations of both code requirements and performance. These experiments not only demonstrated an improvement of up to two orders of magnitude for transmission counts and network initialization time, but also illustrated the minimal increase of code size necessary to provide multi-modal functionality. Finally, we demonstrate the robustness to node compromise provided by the combination of the unbalanced keying scheme and LIGER. In so doing, we have designed, implemented and fully characterized a novel new security architecture for emerging sensor networks.

References

- [1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, August 2002.
- [2] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. *Advances in Cryptology (CRYPTO)*, 740:471–486, 1992.
- [3] S. Capkun and J. Hubaux. BISS: Building Secure Routing out of an Incomplete Set of Security Associations. In *Proceedings of the ACM Workshop on Wireless Security WiSe*, September 2003.
- [4] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2003.
- [5] Crossbow. Wireless sensor networks. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.
- [6] F. Delgosha and F. Fekri. Threshold key-establishment in distributed sensor networks using a multivariate scheme. In *Proceedings of IEEE INFOCOM*, 2006.
- [7] W. Du, J. Deng, S. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2003.
- [8] W. Du, J. Deng, S. Han, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *Proceedings of IEEE INFOCOM*, 2004.

- [9] L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, November 2002.
- [10] A. Fox and S. Gribble. Security on the move: indirect authentication using kerberos. In *Proceedings of the Conference on Mobile Computing and Networking (MobiCom)*, 1996.
- [11] Y. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom)*, 2002.
- [12] J. Hubaux, L. Buttyan, and S. Capkun. Security, testbeds and applications: The quest for security in mobile ad hoc networks. In *Proceedings ACM International Symposium on Mobile ad hoc networking & computing (MobiHoc)*, October 2001.
- [13] J. Kohl and B. Neuman. *The Kerberos Network Authentication Service (V5)*, 1993.
- [14] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.
- [15] D. Liu and P. Neng. Establishing pairwise keys in distributed sensor networks. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2003.
- [16] D. Liu and P. Ning. Location-based pairwise key establishments in static sensor networks. In *Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003.
- [17] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the Conference on Mobile Computing and Networking (MobiCom)*, 2000.
- [18] V. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, and N. Shroff. A minimum cost heterogeneous sensor network with a lifetime constraint. *IEEE Transactions on Mobile Computing*, January 2004.
- [19] M. Miller and N. Vaidya. Leveraging channel diversity for key establishment in wireless sensor networks. In *Proceedings of IEEE INFOCOM*, 2006.
- [20] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21:993–999, 1978.
- [21] P. Papadimitratos and Z. Haas. Secure Routing for Mobile Ad Hoc Networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference*, 2002.
- [22] A. Perrig, R. Canetti, D. Tygar, and D. Song. The tesla broadcast authentication protocol. *RSA CryptoBytes*, 5(2):2–13, 2002.
- [23] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. Spins: Security protocols for sensor networks. *ACM Wireless Networking*, September 2002.

- [24] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer. A Secure routing Protocol for Ad Hoc Networks. In *Proceedings of the IEEE International Conference on Network Protocols*, 2002.
- [25] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28, 1949.
- [26] P. Traynor, G. Cao, and T. La Porta. The Effects of Probabilistic Key Management on Secure Routing in Sensor Networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2006.
- [27] P. Traynor, H. Choi, G. Cao, S. Zhu, and T. La Porta. Establishing pair-wise keys in heterogeneous sensor networks. In *Proceedings of IEEE INFOCOM*, 2006.
- [28] P. Traynor, R. Kumar, H. B. Saad, G. Cao, and T. La Porta. LIGER: A Hybrid Key Management Scheme for Heterogeneous Sensor Networks. In *Proceedings of the ACM/USENIX Fourth International Conference on Mobile Systems Applications and Services (MobiSys)*, 2006.
- [29] P. Traynor, J. Shin, B. Madan, S. Phoha, and T. La Porta. Efficient Group Mobility for Heterogeneous Sensor Networks. In *Proceedings of the IEEE Vehicular Technology Conference (VTC)*, September 2006.
- [30] H. Yang, X. Meng, and S. Lu. Self-organized network layer security in mobile ad hoc networks. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, 2002.
- [31] W. Zhang and G. Cao. Group rekeying for filtering false data in sensor networks: A pre-distribution and local collaboration-based approach. In *Proceedings of IEEE INFOCOM*, 2005.
- [32] S. Zhu, S. Setia, and S. Jajodia. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2003.
- [33] S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing pair-wise keys for secure communication in ad hoc networks: A probabilistic approach. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, November 2003.