LIGER: Implementing Efficient Hybrid Security Mechanisms for Heterogeneous Sensor Networks

Patrick Traynor, Raju Kumar, Hussain Bin Saad, Guohong Cao and Thomas La Porta
Networking and Security Research Center
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
(traynor, rajukuma, binsaad, gcao, tlp)@cse.psu.edu

ABSTRACT

The majority of security schemes available for sensor networks assume deployment in areas without access to a wired infrastructure. More specifically, nodes in these networks are unable to leverage key distribution centers (KDCs) to assist them with key management. In networks with a heterogeneous mix of nodes, however, it is not unrealistic to assume that some more powerful nodes have at least intermittent contact with a backbone network. For instance, an air-deployed battlefield network may have to operate securely for some time until uplinked friendly forces move through the area. We therefore propose LIGER, a hybrid key management scheme for heterogeneous sensor networks that allows systems to operate in both the presence and absence of a KDC. Specifically, when no KDC is available, nodes communicate securely with each other based upon a probabilistic unbalanced method of key management. The ability to access a KDC allows nodes to probabilistically authenticate neighboring devices with which they are communicating. We also demonstrate that this scheme is robust to the compromise of both low and high capability nodes and that the same keys can be used for both modes of operation. Detailed experiments and simulations are used to show that LIGER is a highly practical solution for the current generation of sensors and the unbalanced approach can significantly reduce network initialization time.

Categories and Subject Descriptors

C.2.0 [**Computers-Communication Networks**]: General—*Security and protection*

General Terms

Security, Performance

Keywords

Heterogeneous Sensor Networks, Probabilistic Key Management, Probabilistic Authentication, Hybrid Network Security

MobiSys'06, June 19–22, 2006, Uppsala, Sweden.

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 typically been characterized as being composed of a large number (hundreds to a few thousand [1]) of homogeneous nodes with extreme resource constraints.

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. [13] automatically designates nodes with greater inherent capabilities and energy as cluster heads in order to maximize network lifetime. In Traynor, et al. [19], security is provided by leveraging an unbalanced distribution of symmetric keys. The availability of platforms including the Crossbow Stargate allows such mixed-networks to become a reality.

Security is one of the most difficult problems facing these networks, whether they are composed of a homogeneous or heterogeneous mixture of nodes. As is necessary in most other systems, secure communication is often a mission critical requirement; however, the abovementioned resource constraints of nodes like the Mica2 make the use of solutions including asymmetric cryptosystems unrealistic or impossible in the majority of circumstances. To make matters more complicated, the assumption of physical protection from enemies does not hold for many deployment environments, making the subversion of nodes and their data a realistic threat for these networks. It is therefore extremely important that all attempts at securing these systems, especially in the realm of key management, are as simple and robust as possible.

The currently available schemes for key management 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.

This paper presents LIGER[1], a hybrid key management system for heterogeneous sensor networks. The contributions made in this paper are:

- **Dual-mode Key Management:** We *create an architecture* that allows networks to operate securely in the absence and presence of a KDC. In the absence of a KDC, *we define a stand-alone protocol* that implements the theory in many papers based on probabilistic keying. In the presence of a KDC, we leverage the information deployed to support the stand-alone protocol and preserve least privilege using a protocol better suited for sensor networks than Kerberos. This approach is advantageous as a network can not only always operate in a secure fashion, but also *harness resources as they become available*. If a connection to a KDC is available, the protocol allows nodes to learn enough information so that some level of authentication may be performed if the KDC becomes unavailable for periods of time and the network must operate without a central authority.
- **Probabilistic Authentication:** We develop and test the theory behind using a combination of a subset of a node's keys for authentication. An attractive feature of this scheme is that it requires no more keys than in the stand-alone system and is robust when operating with a KDC. By linking the stand-alone and KDC system, we are able to perform varying levels of authentication of nodes even when the KDC is unavailable for periods of time.
- **Implementation and Analysis:** We implement the LIGER protocol and compare its performance for unbalanced [19] and balanced key distributions [5]. This is the first reported implementation of a probabilistic keying mechanism and sheds light on several important practical issues: it determines bottleneck points in the network, shows the sensitivity of performance to the number of messages required to initiate a network, and highlights the importance of a resilient MAC protocol if high performance is to be attained. We analyze our implementation to quantify the efficiency of implementing a combined protocol. No previous papers have addressed these issues. Additionally, we discuss how the use of the unbalanced key management scheme allows an administrator to customize security policy depending upon the mission of a specific network.

The remainder of this paper is organized as follows: Section 2 discusses related research; Section 3 defines the key establishment protocols in detail including the theoretical foundations of probabilistic authentication; Section 4 discusses the details and savings associated with the implementation of the component schemes; Section 5 explores the performance of these implementations; Section 6 provides concluding comments.

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 [14]. Improvements to this protocol were made in the abundantly used Kerberos [8]. Fox and Gribble proposed the use of Charon [6], a proxy server designed to offload the memory overhead of Kerberos for mobile devices.

A more appropriate centralized keying method for sensor networks is proposed in the SPINS protocol [16], which makes use of a modified version of the TESLA [15] authentication protocol (μTesla). 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 [5]. 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 [2]. 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.

A number of other researchers have proposed incorporating location-based information into the assignment of keys [4, 9, 10]. Because of the expensive and high power-drain characteristics of GPS units, these schemes typically rely upon initial node placement for this information.

The only available key management scheme for heterogeneous sensor networks known to the authors of this paper is discussed by Traynor, et al. [19]. This scheme will be examined in detail in Section 3.1.

None of the previous work has fully defined a protocol or shown implementation results for the security methods they propose. The implementation and measurements presented here show the viabil-

[1]A Liger is the hybrid offspring resulting from the breeding of a lion and a tiger.

ity of these systems in a practical environment and illustrates their sensitivity to MAC layer collisions and network density.

3. 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. The details of the probabilistic authentication scheme will be covered in this section. 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. We differ from most of the previously defined systems by supporting optimizations that allow keys to be deployed in an unbalanced manner, i.e., more keys are deployed in more capable nodes. 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 itself 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). When discussing TIGER we present the details of the probabilistic authentication protocol. We conclude this section by discussing the transition between protocol modes.

Notation

- A, B are principles (e.g. communicating L1 nodes)
- $ID_{A_0}, \cdots, ID_{A_{k-1}}$ 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 $ID_{A_0}, \cdots, ID_{A_{k-1}}$.
- N is a nonce.
- $\{S\}_{\langle K \rangle}$ is a value S encrypted in key K.

3.1 LION: Standalone Key Management

There are a number of advantages to creating wireless sensor networks from a heterogeneous mix of nodes. First, the presence of nodes with additional capabilities greatly reduces the difficulty of implementing secure systems. The ability of the more capable nodes to store extra keying data while incurring only a small relative expense allows for critical memory to be saved on low capability or Level 1 (L1) nodes. Additionally, if high capability or Level 2 (L2) nodes have direct access to an uplink out of the network, very large sensor networks can instead be viewed as a collection of independent, small systems. This significantly decreases the impact of signal blocking barriers (walls, hills, etc).

Based on these arguments, Traynor, et al. [19] propose the deployment of two classes of sensor nodes. L1 (sensor) nodes are very limited in capabilities; L2 nodes are more capable and act as gateways to a backbone network if one is present. L1 and L2 nodes are pre-deployed with k and m keys from a pool of size P, respectively, where $k << m$. The probability of two nodes with different sized key rings sharing at least one key with each other is given in Eq. 1:

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

Given Eq. 1, it is possible to determine network connectivity via a number of trust/communication models. For example, the network administrator may limit data collected by L1 nodes to be backhauled to a wired network via a L2 node. In this case to limit the number of nodes that are involved in key establishment, session keys may only be established by virtue of a direct key match between the L1 node and the L2 node. Alternatively a peer-to-peer model of communication between L1 nodes may be supported by allowing key establishments either directly between two L1 nodes or with the help of a nearby L2 node. This approach is known as the "Limited Trust" communication model as it forces L1 nodes to hold suspicious opinions of their neighbors (for the purpose of key

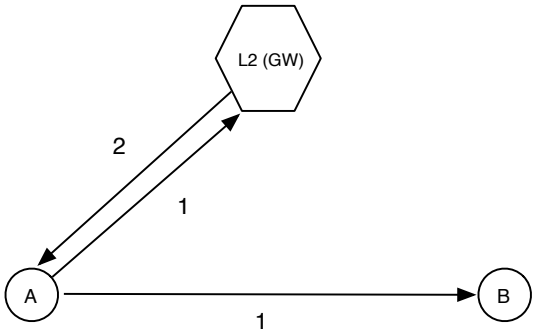

Figure 1: The Direct Key Discovery phase of the LION protocol. First, L1 node A broadcasts out its key identifiers. A neighboring L2 determines it has a match with the L1 and sends a challenge/response message.

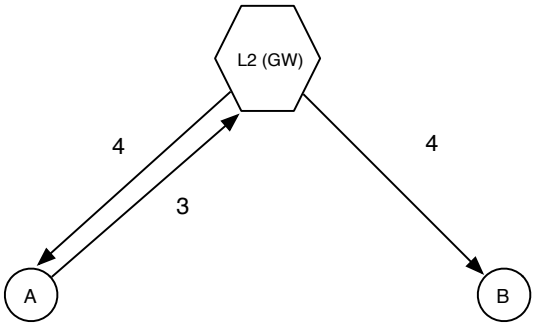

Figure 2: The Indirect phase of the LION protocol. After node A is unable to establish keys with all of its neighbors, it launches a "Request for Assistance" message. The L2, overhearing this request, provides a session key to both parties.

establishment). A number of other, less strict trust relationships including the possibility of allowing up to n intermediate hops for the establishment of session keys have also been proposed. Regardless of the trust/communication model enforced over the network, this unbalanced mechanism of keying allows for the number of keys to remain constant in the L1 nodes while making slight changes to a less resource constrained L2 node. The previously discussed homogeneous network models require all nodes in the network to increase the amount of already limited memory dedicated for key management in order to overlay such schemes.

We use a peer-to-peer model of interaction and therefore select the Limited Trust communication model discussed above. All of the other peer-to-peer models [19] degenerate into this case as the number of keys in L1 nodes (and therefore the probability of directly establishing a key without the use of an L2) decreases significantly.

In the key pre-distribution phase, each of the L1 and L2 nodes receives k or m keys randomly (without replacement for each node) from a pool of size P, respectively.

We now present the message flow for the LION protocol. This protocol supports both unbalanced and balanced key distribution. After deployment, an L1 node learns its neighbors through the exchange of `Hello` messages, and then attempts to establish keys with its neighbors. 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 [5], this method does not compromise the integrity of the node itself. If a neighboring node overhears this transmission and determines that it shares one of the keys associated with the broadcast, it responds to the source with a challenge/response. In Figure 1 we 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) $A \rightarrow * : A, N, ID_{A_0}, \cdots, ID_{A_{k-1}}$
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) $L2 \rightarrow * : A, B, N, \{K_{A,B}, N\}_{\langle K_{A,L2} \rangle}, \{K_{A,B}, N\}_{\langle K_{B,L2} \rangle}$

Figure 2 shows the indirect phase of the protocol with the "Request for Assistance" message being transmitted to the neighboring L2 node in accordance with the Limited Trust model. This message would instead 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.

Authentication of neighbors is accomplished by challenging an adjacent node on multiple shared keys. This scheme is discussed in detail in Section 3.2.3.

A criticism of pre-distribution schemes is that copies of each key are stored in multiple nodes, therefore making the system less secure. However, with keys distributed in a uniform random fashion over a network of 1,000 nodes (12.5% being L2s) [19], each key is likely to be located in approximately 1% of the nodes on average. Because the majority of keys stored in each node are unused throughout the network [5], the compromise of a single node's keys is not equivalent to the loss of the same number of actively used keys. Regardless, in order to locate a specific key, an attacker would have to physically compromise almost 100 nodes (12.5% being L2s). 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. We therefore assert that this mechanism is appropriate for key management in sensor networks.

3.2 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.

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 protocol definition and analyze the robustness of the authentication mechanism.

3.2.1 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 if a KDC is present. 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. A discussion of this type of authentication is given in Section 3.3. Below we provide the discussion for the method when applied to operation with a KDC.

A similar concept, called the q-composite scheme [2], has been previously proposed. The q-composite scheme is designed to improve the robustness of probabilistic keys, not to provide authentication. As mentioned before, with the q-composite scheme nodes must share at least q keys in common in order to establish a connection. While it provides additional security for the system, significantly more keys are required as q increases.

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 a simple XOR on the selected q keys. This operation, chosen for its speed and ubiquity across all platforms, is guaranteed to create an unguessable, pseudo-random value from the key space as demonstrated by Shannon [17]. 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 Section 3.2.3.

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. In our system, when using a KDC 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 small as shown in Section 3.2.3.

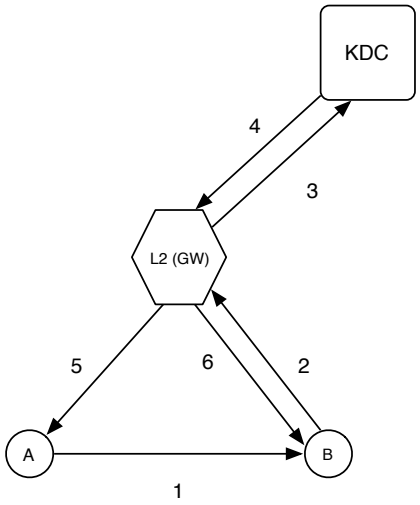

Figure 3: The TIGER message flow for key establishment between L1 nodes. Node A sends a token to Node B indicating its desire to establish a session key. B includes its own token and forwards that message to the KDC via the gateway. The KDC determines the validity of both tokens and returns a session key to both parties. A copy of this key is encoded in one of the allowable authenticator keys from both A and B.

3.2.2 TIGER Protocol Definition

TIGER strives to take 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 a neighborhood 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.

In order to minimize the effects of an L2 node being compromised, we restrict the L1 nodes with which an L2 can establish keys to those directly within its transmission range. This least privilege is accomplished with a token mechanism described below.

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 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 [12, 20], 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

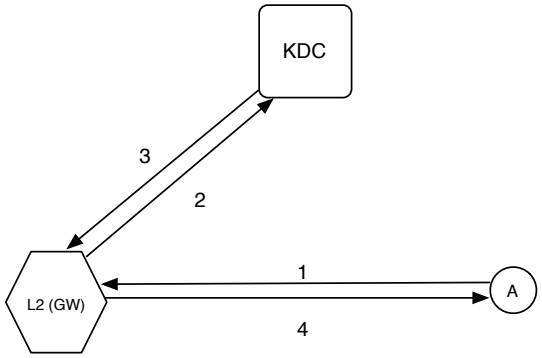

Figure 4: The TIGER message flow for establishing keys between L1 and L2 nodes. Because the L2 node is required to include a token generated by the L1 with which it is trying to establish a session, least privilege is enforced.

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 3 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)$L2 \rightarrow KDC$: Forward Message 2 to KDC
4)$KDC \rightarrow L2 : A, B, N, MAP_{A'}, \{K_{A,B}, N\}_{\langle K_{A'\_AUTH}\rangle}$,
$MAP_{B'}, \{K_{A,B}, N\}_{\langle K_{B'\_AUTH}\rangle}, MAC(K_{A,B}, A|B|N|K_{A,B})$
5)$L2 \rightarrow A : A, B, N, MAP_{A'}, \{K_{A,B}, N\}_{\langle K_{A'\_AUTH}\rangle}$,
$MAC(K_{A,B}, A|B|N|K_{A,B})$
6)$L2 \rightarrow B : A, B, N, MAP_{B'}, \{K_{A,B}, N\}_{\langle K_{B'\_AUTH}\rangle}$,
$MAC(K_{A,B}, A|B|N|K_{A,B})$

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 if keys are gradually compromised by an adversary.

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. Additionally, TIGER, like Kerberos, 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; 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 being within physical proximity of them, least privilege is preserved. The messages to implement this protocol follow the flow shown in Figure 4 and use the format below:

1) $A \rightarrow L2 : A, L2, N, MAP_A, MAC(K_{A\_AUTH}, L2|N|MAP_A)$
2) $L2 \rightarrow KDC$: 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) $L2 \rightarrow A : A, N, MAP_{A'}\{K_{A,L2}, N\}_{\langle K_{A'\_AUTH}\rangle}$,
$MAC(K_{A,L2}, A|N|MAP_{A'}, \{K_{A,L2}, N\}_{\langle K_{A'\_AUTH}\rangle})$

3.2.3 Analysis

In this subsection we discuss the robustness of the probabilistic authentication between the KDC and an L1 node.

Through Eqs 2 and 3, we determine the probability that two nodes with different numbers of keys (k and m, respectively) share at least q keys in common with each other. Our equation degenerates into the q-composite equation [2] when $k = m$.

Let $p(i)$ be the probability that two nodes share exactly i keys in common. The number of ways in which a key ring of size k and one of size m can be chosen from a pool P are $\binom{P}{k}$ and $\binom{P}{m}$, respectively. There are also $\binom{P}{i}$ ways in which two nodes can chose i keys in common. After i common keys have been selected, there remains $(m-i)+(k-i)$ key rings that must still be constructed from the remaining pool $(P-i)$. The number of ways to distribute these remaining keys between key rings of size k and m is $\binom{(m-i)+(k-i)}{m}$. The probability that two nodes share exactly i keys in common is:

$$p(i) = \frac{\binom{P}{i}\binom{P-i}{(m-i)+(k-i)}\binom{(m-i)+(k-i)}{m-i}}{\binom{P}{m}\binom{P}{k}} \quad (2)$$

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

$$1 - \sum_{i=0}^{q-1} p(i) \quad (3)$$

Figure 5 demonstrates the probability, given a pool of 10,000 keys and L1 nodes containing 10, 20 or 30 keys, that an adversary with a varying number of the keys from P would contain all k keys

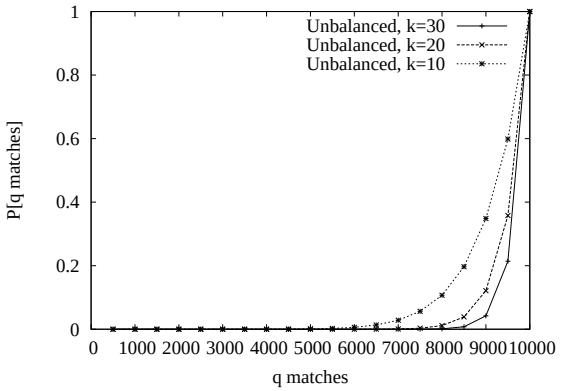

Figure 5: The probability that an L1 node can be cloned given a varying number of compromised keys.

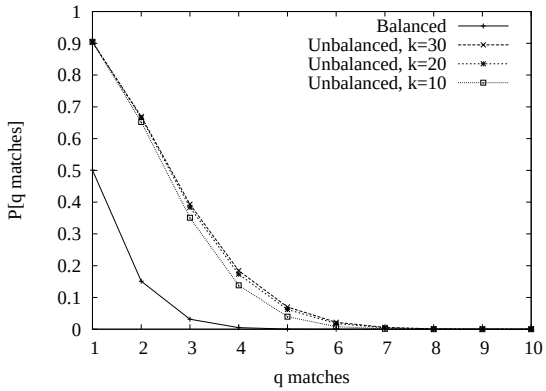

Figure 6: The probability that two nodes can match q keys with each other for the balanced (83 keys/node) and unbalanced (30 keys/L1; 750 keys/L2) cases. Notice that L1 nodes have a much higher probability of matching multiple keys with their neighbors for all values of q due to the presence of L2 nodes.

stored in a random L1 node. Approximately 63%, 79% and 86% of the entire key pool must be compromised before an attacker has even a 1% chance of successfully impersonating an L1 node for k = 10, 20 and 30, respectively.

Figure 6 shows the probability of an L2 node deployed with 750 keys [19] sharing at least q keys with any L1. This graph demonstrates that an L2 node is highly unlikely to match enough keys to impersonate a specific L1 node, thus gaining its session keys, when the KDC can be reached. For example, consider an L1 node pre-deployed with 30 keys, 10 of which are used to construct the authenticator key. In this case, the probability that a L2 node can impersonate the L1 node is below $2.52 \cdot 10^5$. This probability falls to $9.37 \cdot 10^8$ if a L1 is predeployed with 10 keys, all of which are used to construct the authenticator key.

3.3 LIGER: 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: A 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 in to 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 6, 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 [22], that the

Table 1: L1 code size on Mica2 motes

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

Table 2: Functions used in L1 Node in each scheme

Function Name	LION	TIGER	LIGER
addToSendList()	✓	✓	✓
sendFromList()	✓	✓	✓
hasKeyID()	✓	✓	✓
searchNode()	✓	✓	✓
getKeyFromID()	✓	✓	✓
insertNode()	✓	✓	✓
insertIndirNode()	✓		✓
deleteIndirNode()	✓		✓
initializeKeys()	✓	✓	✓
LionDirectRep()	✓		✓
LionIndirectRep()	✓		✓
LionIndirectSearch()	✓		✓
LionDirectSearch()	✓		✓
TigerSendReqToGW()		✓	
LigerSendReqToGW()			✓
LigerSendReqToDst()			✓
receiveMsg()	✓	✓	✓

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 [5]; 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 [23] 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

Table 3: GW functions for TIGER and LIGER schemes

Function Name	Tiger	LIGER
open_serial()	✓	✓
calcrc()	✓	✓
read_packet()	✓	✓
write_packet()	✓	✓
getReplyFromKDC()	✓	✓
TigerHandleReq()	✓	
LigerHandleReq()		✓
readPublicPrivateKeys()	✓	✓
threadMain()	✓	✓
main()	✓	✓

Table 4: KDC functions for TIGER and LIGER schemes

Function Name	Tiger	Liger
readGWPublicKey()	✓	✓
generateSessionKey()	✓	✓
TigerHandleReq()	✓	
verifyLigerHMAC()		✓
LigerHandleReq()		✓
threadMain()	✓	✓
main()	✓	✓

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.

4. DESIGN AND IMPLEMENTATION

4.1 Experimental Parameters

In this section, we discuss the implementation of the LION, TIGER and LIGER components for our hybrid key management system. The Crossbow Mica2 mote [3], 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.

We chose RC5 as the symmetric key cipher for this system due to the fact that it has already been implemented and tested for the Mica2 platform in TinySec [7]. An implementation is also available in the OpenSSL cryptographic library, making it the most suitable algorithm for symmetric keying operations for current implementations of secure sensor systems. RSA using 1024-bit keys from the same libraries was also used in order to implement secure communication between the L2 and KDC nodes. The implementation of the protocols on the L1 nodes occupied approximately 19, 16

Table 5: Size (in bytes) for various applications, with and without LIGER for Mica2 motes

Application	Size in ROM	Projected size in ROM with LIGER	Size in RAM	Projected size in RAM with LIGER
CntToLedsAndRfm	10948	31930	447	979
OscilloscopeRF	11828	32810	504	1036
SenseToRfm	11562	32544	465	997
TinyDBApp	62854	83836	2828	3360

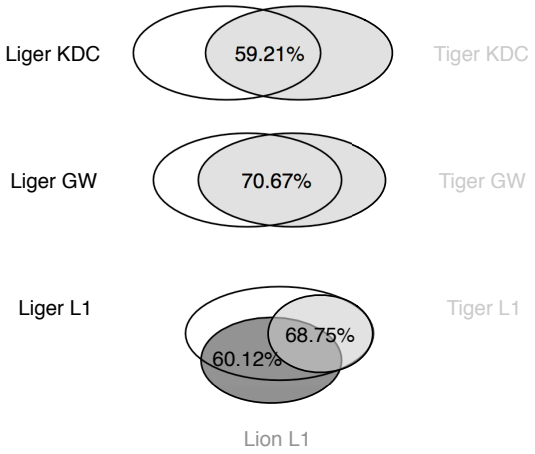

Figure 7: Functionality overlap and code reuse for components for LION, TIGER and LIGER schemes. The percentage values represent the number of lines of code that are reused between two modes.

and 21 KB of ROM and 500 bytes of RAM for LION, TIGER and LIGER, respectively (see Table 1).

4.2 Software Design Issues

Providing sensor networks with an architecture that allows them to take advantage of resources as they become available increases the robustness of such systems. Because these devices are so constrained, however, the cost of flexibility must be carefully considered. We therefore examine the efficiency of implementing such flexibility for security mechanisms in this environment. Tables 2, 3 and 4 enumerate the functions used in LION, TIGER and LIGER in the various schemes in which they are employed. Below we discuss the implementation of L1 nodes, L2 nodes and the KDC, communication limitations, and system integration.

L1 Node Implementation: Table 2 shows the functions used in the LION, TIGER and LIGER schemes in L1 nodes. In spite of LION and TIGER begin fundamentally different means of establishing keys with neighbors, the implementation of both of these systems share a large number of common functions. Implemented as independent schemes, LION and TIGER binaries require 19,636 and 16,148 bytes of program memory and 522 and 458 bytes of RAM, respectively. However, when combined as LIGER, the total space required to store the security mechanisms on an L1 nodes is less than 1KB more than LION and approximately 4KB more for TIGER (see Table 1). Careful design and extensive reuse of code make the implementation of LIGER take approximately 58% of the program memory and 54% of the RAM required for the two modes implemented alone (see Figure 7). Because the commercially available platforms such as the Mica2 are limited to 128KB of program memory and 4KB of RAM [3], such efficiency is necessary for any security solution in this environment.

The most important design consideration in terms of code size is the effect any security solution has upon the ability of a sensor node to perform its primary mission. Simply stated, any implementation of a security scheme that does not leave space for real applications to run is in fact not a valid security solution for this environment. We therefore compare the footprint of LIGER against the space required for a number of commonly used applications for sensor networks. Table 5 lists a number of programs provided with the standard installation of TinyOS and others that have been independently distributed. Because of the small size of LIGER, programs ranging in size from CntToLedsAndRfm [18] to TinyDB [11] can be implemented securely on the constrained Mica2 platform.

L2 node and KDC Implementation: As shown in Tables 3 and 4, gateways and KDCs have different functions responsible for the initial processing of messages in TIGER and LIGER modes. While memory use is much less of a concern when compared to the L1 platform, the combination of LION and TIGER schemes provides an extremely efficient implementation. As is demonstrated in Figure 7, approximately 70% and 59% of the lines of code are shared between these two modes in the gateway and KDC platforms, respectively. Because the gateway and KDC nodes are built on different architecture, a direct comparison of the binaries is not appropriate. In our implementation, the KDC and the GW have both TIGER and LIGER capabilities. Based on the type of packet received from L1 nodes, the gateway forwards the request to the KDC, and then sends the reply to the L1 node(s). The KDC also responds to requests from the gateway based on the type of the request. Hence the key establishment scheme to be used is entirely decided by the request type originating at L1 nodes.

Communication Constraints: One of the major constraints on implementing any scheme on a sensor platform is the small available payload size of packets. Under TinyOS specifically, this limitation is set to 29 bytes. Accordingly, all of the wireless interactions between nodes in the LION, TIGER and LIGER schemes must adhere to this restriction. The sizes of all of the components are therefore carefully considered. Each of the symmetric keys deployed in L1 nodes are 8 bytes, which matches the key size used for the TinySec implementation of the RC5 block cipher [7]. Both node identifiers and nonces occupy 2 bytes each. We believe that 2 bytes is sufficient for both fields because it permits the network size to be extremely large and also provides sufficient protection against reused nonces due to current battery lifetimes. Authentication is provided by CBC residue and occupies a total of 4 bytes, as provided by TinySec. While not appropriate for other environments, an online attack of this authentication mechanism would require an average attack span of 20 months because of the limited bandwidth in this setting [7].

Integration: One of the major contributions of this work is the ability of this system to switch between standalone and infrastructure-supported modes. In so doing, we allow for our network to take advantage of the greatest available resources to help with the process of establishing security associations between nodes. Accordingly, nodes will always begin their attempts to establish session keys with their neighbors under the infrastructure-supported mode

Table 6: Microbenchmark results for LION and TIGER modes.

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

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

of LIGER. The advantages to starting in this mode include not only harnessing the resources of a KDC, but also not revealing any information about the key identifiers stored in each L1 mode. If no backhaul link to a KDC is available, nodes then default to LION mode and attempt to establish keys with their neighbors. If a link can be established with a KDC at some future point, nodes can opt to rekey with their neighbors (selectively or in total). While the L2 node can provide a much weaker authentication of L1 nodes because it likely shares a small set of keys with each of its neighbors, nodes that need more concrete guarantees can force their neighbors to be verified by a central authority as soon as this link can be established.

The combination of LION and TIGER schemes therefore yields not only a robust and efficient system in terms of security, but it also can be implemented in a manner that is not overly burdensome on constrained platforms.

5. PERFORMANCE EVALUATION

In the following subsections we present 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.

5.1 Node Benchmarks

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 6.

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 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.

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.

Because 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 $\alpha = 20\ msec$ in our simulations because this provides sufficient spacing for TOSSIM while still allowing for maximum channel utilization.

In the simulations, we pre-deploy a sufficient number of keys in each node to provide 0.99999 network connectivity as described in Section 3.1. We consider a network 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 node has 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 discussed in Table 6, real Stargate nodes will achieve much lower processing delays than the Mica2 motes, so our results in this section 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

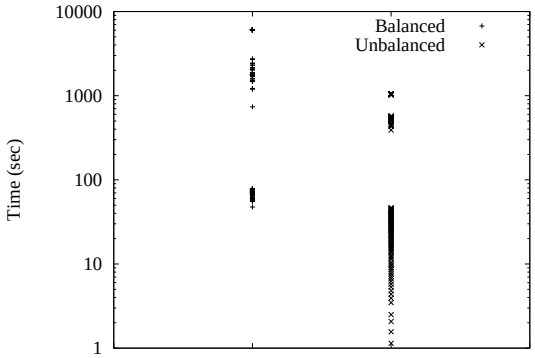

Figure 8: The termination of individual nodes in the network for both the direct (lower clusters) and indirect (upper clusters) phases of initialization. Subsequent phases are spaced at 400 seconds.

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 that there is no compromise during network initialization, as is often done in this area of research [21, 22], 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 random 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.

Table 7: 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

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 discernable. 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 graph denotes a node completing the phase by achieving at least 90% connectivity (key matching) with its neighbors.

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. Using short timers increases the number of nodes competing for the medium and therefore adds the potential for more collisions. With increased collisions comes the need to launch additional rounds of the indirect phase in the future. Setting timers too conservatively, as was purposely done to determine the lifetime of each phase in Figure 8, 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 standard deviation (σ) of the termination time. Based on experiments with several values of timer, we choose the average time plus 0.5σ as the timer value.

Table 7 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 9 shows the network initialization times for varying node densities for both the balanced and unbalanced (5 L2 gateways) cases. 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 net-

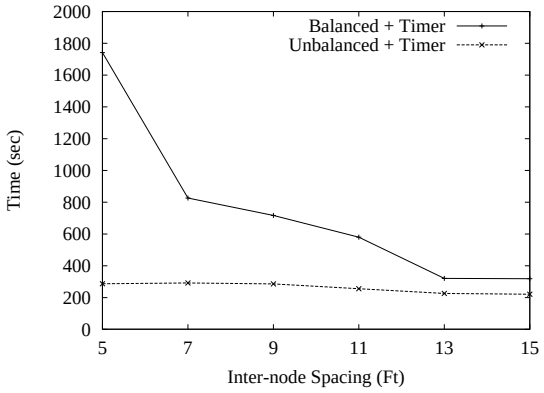

Figure 9: Balanced and Unbalanced network initialization times for the LION scheme with varying inter-node spacing.

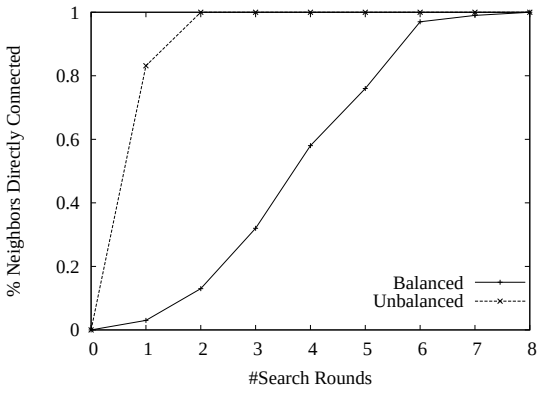

Figure 10: The percentage of nodes with 90% connectivity directly to their neighbors across a number of key establishment rounds.

work, random graph theory [5] 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 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. The cumulative distribution function of the network initialization points, shown for the five feet spacing case in Figure 10, therefore allows the number of indirect key establishment rounds to be picked depending upon the administrator's guidelines for direct connectivity.

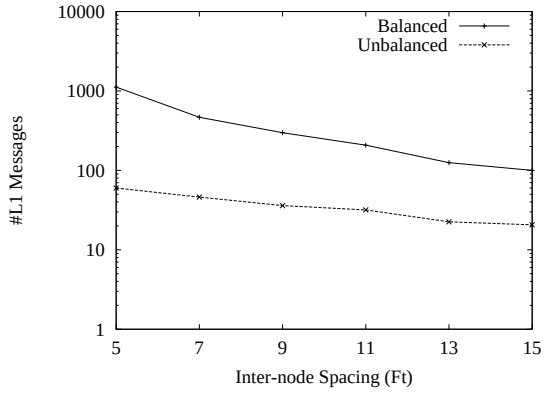

Figure 11: The average number of messages per L1 node to initialize networks with varying inter-node spacing.

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. Because transmission bandwidth is limited (38.4 kbps theoretical maximum), the sheer number of packets needed to establish secure connections is overwhelmingly the source of delay in the system. This problem is not realistically solved simply by the introduction of higher bandwidth radios such as those included with the new MICAz [3]. 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 is such that certain nodes throughout the network, specifically L2s, are expected to process an elevated level of packets compared to their neighbors. Table 7 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.

5.3 Single Node Initialization

In addition to network initialization time, we determined the amount of time for a single node to establish key matches with 95% of its neighbors in the five feet spacing case. This is important for cases in which a sensor node is added to the field after the network is in operation, perhaps to increase density or replace a failed node.

For the unbalanced case, a single L1 node was able to reach this goal in a single round of direct and indirect searching. The total time required was 5.526 seconds. For the balanced case, a node required one direct and two indirect searches for a total time of 143.435 seconds to achieve the same connectivity. This illustrates another benefit of using unbalanced key distribution.

6. CONCLUSION

To the best of our knowledge, this is the first paper to address hybrid key management issues in heterogeneous sensor networks.

Networks can be made more robust when they leverage all available resources. It is for this reason that we have presented LIGER, a hybrid key management scheme for heterogeneous sensor networks. We have demonstrated that this system can be efficiently implemented to not only take advantage of the presence or absence of a KDC without the need for additional key storage, but also to reuse functionality to require a minimal footprint for this robustness. Furthermore, we have shown that the probabilistic method of authentication is robust to a variety of situations including a high number of node compromises.

Through performance analysis, we demonstrate the savings inherent to the unbalanced key management scheme. We also show that the use of different cryptographic algorithms can affect the composition of a network for reasons of economics. Most importantly, we demonstrate that the use of a hybrid key management scheme in heterogeneous sensor networks is practical, robust and customizable to varying mission constraints.

In this work we determined efficient timer settings experimentally to show the utility of the LIGER system and unbalanced key distribution. In our experiments we found that the average phase completion time was tightly coupled to the node density. One possible approach to dynamically setting interphase timers is for nodes to estimate the network density by counting neighbors, and then having each node set its own timer. We will explore such solutions in further work.

7. ACKNOWLEDGMENTS

We would like to thank our reviewers, anonymous or otherwise, who helped us with their insightful and constructive comments. We also extend a special thanks to Mahadev Satyanarayanan for his assistance during the shepherding process.

This work was supported in part by The Technology Collaborative (TTC), Army Research Office (W911NF-05-1-0270) and the National Science Foundation (CNS-0524156 and CNS-0519460). Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of The Technology Collaborative, Army Research Office, or National Science Foundation.

8. REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, August 2002.

[2] 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.

[3] Crossbow. Wireless sensor networks. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.

[4] W. Du, J. Deng, S. Han, and P.K. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *Proceedings from the Conference of the IEEE Communications Society (Infocom)*, 2004.

[5] L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, 2002.

[6] 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.

[7] C. Karlof, N. Sastry, and D. Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *Proceedings of the ACM Conference of Embedded Networked Sensor System (SenSys)*, 2004.

[8] J. Kohl and B. Neuman. *The Kerberos Network Authentication Service (V5)*, 1993.

[9] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, 2003.

[10] 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.

[11] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 30(1):122–173, 2005.

[12] S. Marti, T.J. 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.

[13] 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.

[14] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Commuinications of the ACM*, 21:993–999, 1978.

[15] A. Perrig, R. Canetti, D. Tygar, and D. Song. The tesla broadcast authentication protocol. *RSA CryptoBytes*, 5(2):2–13, 2002.

[16] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. Spins: Security protocols for sensor networks. *ACM Wireless Networking*, September 2002.

[17] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28, 1949.

[18] TinyOS. http://www.tinyos.net, 2005.

[19] Patrick Traynor, Heesook Choi, Guohong Cao, Sencun Zhu, and Thomas La Porta. Establishing pair-wise keys in heterogeneous sensor networks. In *Proceedings of IEEE INFOCOM*, 2006.

[20] 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.

[21] W. Zhang and G. Cao. Group rekeying for filtering false data in sensor networks: A predistribution and local collaboration-based approach. In *Proceedings from the Conference of the IEEE Communications Society (Infocom)*, 2005.

[22] S. Zhu, S. Setia, and S. Jajodia. Leap: Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, 2003.

[23] 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*, 2003.