A Key-Management Scheme for Distributed Sensor Networks*

Laurent Eschenauer
Electrical and Computer Engineering Department
University of Maryland
College Park, MD, USA
laurent.e@mail.com

Virgil D. Gligor
Electrical and Computer Engineering Department
University of Maryland
College Park, MD, USA
gligor@eng.umd.edu

ABSTRACT

Distributed Sensor Networks (DSNs) are ad-hoc mobile networks that include sensor nodes with limited computation and communication capabilities. DSNs are dynamic in the sense that they allow addition and deletion of sensor nodes after deployment to grow the network or replace failing and unreliable nodes. DSNs may be deployed in hostile areas where communication is monitored and nodes are subject to capture and surreptitious use by an adversary. Hence DSNs require cryptographic protection of communications, sensor-capture detection, key revocation and sensor disabling. In this paper, we present a key-management scheme designed to satisfy both operational and security requirements of DSNs. The scheme includes selective distribution and revocation of keys to sensor nodes as well as node re-keying without substantial computation and communication capabilities. It relies on probabilistic key sharing among the nodes of a random graph and uses simple protocols for shared-key discovery and path-key establishment, and for key revocation, re-keying, and incremental addition of nodes. The security and network connectivity characteristics supported by the key-management scheme are discussed and simulation experiments presented.

Keywords

key management, sensor networks, random graphs, probabilistic key sharing

*This work was supported in part by the U.S. Army Research Office under Award No. DAAD19-01-1-0494, and by the U.S. Army Research Laboratory under Cooperative Agreement DAAD19-01-2-0011 for the Collaborative Technology Alliance for Communications and Networks.

CCS'02, November 18–22, 2002, Washington, DC, USA.

1. INTRODUCTION

Distributed Sensor Networks (DSNs) share several characteristics with the more traditional embedded wireless networks [13]. Both include arrays of sensor nodes that are battery powered, have limited computational capabilities and memory, and rely on intermittent wireless communication via radio frequency and, possibly, optical links. Both include data-collection nodes, which cache sensor data and make it available for processing to application components of the network, and control nodes, which monitor the status of and broadcast simple commands to sensor nodes. Although in both networks most nodes have limited, if any, mobility after deployment, some nodes are highly mobile (e.g., data collection and control nodes placed on humans, vehicles, aircraft). However, DSNs differ from the traditional embedded wireless networks in several important areas, namely: their scale is orders of magnitude larger than that of embedded wireless networks (e.g., tens of thousands as opposed to just tens of sensor nodes); they are dynamic in the sense that they allow addition and deletion of sensor nodes after deployment to extend the network or replace failing and unreliable nodes without physical contact; and they may be deployed in hostile areas where communication is monitored and sensor nodes are subject to capture and manipulation by an adversary. These challenging operational requirements place equally challenging security constraints on DSN design. (For a detailed analysis of the operational and security constraints of DSNs, the reader is referred to the work of Carman, Kruus, and Matt [3]).

Communication Security Constraints. The capabilities of the sensor nodes for large-scale DSNs range from those of Smart Dust sensors [5, 9] that have only 8Kb of program and 512 bytes for data memory, and processors with 32 8-bit general registers that run at 4 MHz and 3.0V (e.g., the ATMEL 90LS8535 processor), to sensors that are over an order of magnitude more capable in terms of processing speed (e.g., the MIPS R4000 processors) and memory capacity. The power, energy and the related computational and communication limitations of nodes in this range make it impractical to use typical asymmetric (public-key) cryptosystems to secure communications. For example, Carman, Kruus, and Matt [3] report that on a mid-range processor, such as the Motorola MC68328 "DragonBall," the energy consumption for a 1024-bit RSA encryption (signature) operation is much higher than that for a 1024-bit AES encryption operation; i.e., about 42 mJ (840 mJ) versus 0.104 mJ. Further, the energy consumption for transmitting a 1024-bit block over a distance of approximately 900 meters using a typical communication subsystems such as Sensoria WINS NG RF at 10 Kbps and 10 mW of power is about half that of RSA encryption (i.e., 21.5 mJ) and even less for reception (14.3 mJ). Substantially less energy is spent to communicate over smaller distances, since power is proportional to the square of the distance. Also, in the range of sensor capabilities we consider, symmetric-key ciphers and hash functions are between two to four orders of magnitude faster than digital signatures [3]. Hence, symmetric-key ciphers, low-energy, authenticated encryption modes [6, 8, 11], and hash functions become the tools of choice for protecting DSN communications.

Key Management Constraints. Traditional Internet style key exchange and key distribution protocols based on infrastructures using trusted third parties are impractical for large scale DSNs because of the unknown network topology prior to deployment, communication range limitations, intermittent sensor-node operation, and network dynamics. To date, the only practical options for the distribution of keys to sensor nodes of large-scale DSNs whose physical topology is unknown prior to deployment would have to rely on *key pre-distribution.* Keys would have to be installed in sensor nodes to accommodate secure connectivity between nodes. However, traditional key pre-distribution offers two inadequate solutions: either a single *mission key* or a set of separate *n-1 keys*, each being pair-wise privately shared with another node, must be installed in every sensor node.

The single mission-key solution is inadequate because the capture of *any* sensor node may compromise the entire DSN since selective key revocation is impossible upon sensor-capture detection. In contrast, the pair-wise private sharing of keys between every two sensor nodes avoids wholesale DSN compromise upon node capture since selective key revocation becomes possible. However, this solution requires pre-distribution and storage of $n-1$ keys in each sensor node, and $n(n-1)/2$ per DSN, which renders it impractical for DSNs using, say, more than 10,000 nodes, for both intrinsic and technological reasons. First, pair-wise private key sharing between any two sensor nodes would be unusable since direct node-to-node communication is achievable only in small node neighborhoods delimited by communication range and sensor density. Second, incremental addition and deletion as well as re-keying of sensor nodes would become both expensive and complex as they would require multiple keying messages to be broadcast network-wide to all nodes during their non-sleep periods (i.e., one broadcast message for every added/deleted node or re-key operation). Third, a dedicated RAM memory for storing $n-1$ keys would push the on-chip, sensor-memory limits for the foreseeable future, even if only short, 64-bit, keys are used[1] and would complicate fast key erasure upon detection of physical sensor tampering (viz., Section 2.4).

Our Approach. We propose a simple key pre-distribution scheme that requires memory storage for only few tens to a couple of hundred keys, and yet has similar security and superior operational properties when compared to those of the pair-wise private key-sharing scheme. Our scheme relies on probabilistic key sharing among the nodes of a random graph and uses a simple shared-key discovery protocol for key distribution, revocation and node re-keying. Prior to DSN deployment, we distribute a ring of keys to each sensor node, each key ring consisting of randomly chosen k keys from a large pool of P keys, which is generated off-line. Because of the random choice of keys on key rings, a shared key may not exist between some pairs of nodes. Although a pair of nodes may not share a key, if a path of nodes sharing keys pair-wise exists between the two nodes at network initialization, the pair of nodes can use that path to exchange a key that establishes a direct link. Therefore, full shared-key connectivity offered by pair-wise private key sharing between every two nodes becomes unnecessary. We use random graph analysis and simulation to show that what really matters in key pre-distribution is the shared-key connectivity of the resulting network. For example, we show that to establish "almost certain" shared-key connectivity for a 10,000-node network, a key ring of only 250 keys have to be pre-distributed to every sensor node where the keys were drawn out of a pool of 100,000 keys, leaving a substantial number of keys available for DSN expansion (viz., Sections 3 and 4). We also show that the security characteristics of probabilistic key distribution based on random graphs are suitable for solving other key-management problems of DSNs, such as selective revocation of a node's keys, node re-keying, and incremental addition/deletion of nodes.

Related Work. Symmetric key pre-distribution has been used in past research, but with a focus on group and broadcast communication. For group communication [1, 2], this research tries to accommodate any set of up to k users while being secure against collusion between some of them. Pre-distribution is used to alleviate the cost of communication between group members and to setup a common secret key; however, memory constraints are not placed on group members. Other work on broadcast encryption [4] focuses on key distribution to support broadcast communication between slave nodes and a master node - an impractical approach for large-scale DSNs.

2. OVERVIEW OF THE BASIC SCHEME

In this section, we present the basic features of our scheme, deferring its analysis for the next section.

2.1 Key Distribution

In our scheme, key distribution consists of three phases, namely key pre-distribution, shared-key discovery, and path-key establishment.

The *key pre-distribution phase* of our scheme consists of five off-line steps, namely generation of a large pool of P keys (e.g., 2^{17} - 2^{20} keys) and of their key identifiers; random drawing of k keys out of P without replacement to establish the key ring of a sensor; loading of the key ring into the memory of each sensor; saving of the key identifiers of a key ring and associated sensor identifier on a trusted controller node; and for each node, loading the i-th controller node with the key shared with that node. [2] As shown in the next

[1] The approximately 80KB of dedicated key memory will have to be stored in RAM since keys can be dynamically added/deleted. This represents a substantial fraction of the on-chip RAM memories for the processors at the high end of the range considered.

[2] Note that the key shared by a node with the i-th controller node, K^{ci}, can be computed as $K^{ci} = E_{K_x}(ci)$, where $K_x = K_1 \oplus, ..., \oplus K_k$, K_i are the keys of the node's key ring, ci is the controller's identity, and E_{K_x} denotes encryption

section, the key pre-distribution phase ensures that only a small number of keys need to be placed on each sensor node's key ring to ensure that any two nodes share (at least) a key with a chosen probability; e.g., for a probability of 0.5, only 75 keys drawn out of a pool of 10,000 keys need to be on any key ring.

The *shared-key discovery* phase takes place during DSN initialization in the operational environment where every node discovers its neighbors in wireless communication range with which it shares keys. The simplest way for any two nodes to discover if they share a key is that each node broadcast, in clear text, the list of identifiers of the keys on their key ring. This approach does not give an adversary any attack opportunity that he does not already have. For example, if an adversary captures a node he can discover which key of that node is used for which link by decrypting communications; and if he does not capture a node, the adversary can mount a traffic analysis attack in the absence of key identifiers.

Alternate methods exist which hide key-sharing patterns among nodes from an adversary thereby establishing *private* shared-key discovery. These methods would force an adversary to conduct traffic analysis to discover the pattern of key sharing. For example, for every key on a key ring, each node could broadcast a list $\alpha, E_{K_i}(\alpha), i = 1, \cdots, k$, where α is a challenge. The decryption of $E_{K_i}(\alpha)$ with the proper key by a recipient would reveal the challenge α and establish a shared key with the broadcasting node.

The shared-key discovery phase establishes the topology of the sensor array as seen by the routing layer of the DSN. A *link* exists between two sensor nodes only if they share a key; and if a link exists between two nodes, all communication on that link is secured by link encryption. Note that it is possible that the same key is shared by more than a pair of sensor nodes, since the key rings consist of keys drawn randomly from the same pool. This does not cause a link-security exposure because, in normal mode of operation sensor nodes trust each other and, during the revocation phase following node-capture detection, revocation of a captured node's key ring ensures that the small set of (k) keys on that ring are removed network-wide.

The *path-key establishment* phase assigns a *path-key* to selected pairs of sensor nodes in wireless communication range that do not share a key but are connected by two or more links at the end of the shared-key discovery phase. Path keys need not be generated by sensor nodes. The design of the DSN ensures that, after the shared-key discovery phase is finished, a number of keys on a key ring are left unassigned to any link. For example, both analysis (Section 3) and simulations (Section 4) show that even without special provisioning a substantial number of keys are left unused on key rings. Provisioning for sufficient ring keys that are left unassigned by the determination of key-ring size (k) can also anticipate both the effects of revocation and those of incremental addition of new sensor nodes, since both may require the execution of the path key establishment phase after shared-key discovery. The analysis and simulations presented in the next sections indicates that such provisioning is especially simple.

with node key K_x. Hence, the keys shared by a node with controllers, which are only infrequently used, need not take any space on the key ring. However, in this case, a K^{ci} would change upon any key change on a ring.

2.2 Revocation

Whenever a sensor node is compromised, it is essential to be able to revoke the entire key ring of that node. To effect revocation, a controller node (which has a large communication range and may be mobile) broadcasts a single revocation message containing a signed list of k key identifiers for the key ring to be revoked. To sign the list of key identifiers, the controller generates a signature key K_e and unicasts it to each node by encrypting it with a key K^{ci}. (Recall that keys K^{ci} are shared by the i-th controller with each sensor node during key pre-distribution phase.)

After obtaining the signature key, each node verifies the signature of the signed list of key identifiers, locates those identifiers in its key ring, and removes the corresponding keys (if any). Once the keys are removed from key rings, some links may disappear, and the affected nodes need to reconfigure those links by restarting the shared-key discovery and, possibly path-key establishment, phase for them. Because only k out of P keys are removed from the pool for every revoked node, revocation affects only a few other nodes and a small part of their key ring but it disables all connectivity of the compromised node.

2.3 Re-Keying

Although it is anticipated that in most DSNs the lifetime of a key shared between two nodes exceeds that of the two nodes, it is possible that in some cases the lifetime of keys expires and re-keying must take place. Re-keying is equivalent with a self-revocation of a key by a node. As such, it does not involve any network-wide broadcast message from a controller and, hence, is especially simple. After expired-key removal, the affected nodes restart the shared-key discovery and, possibly, the path-key establishment, phase.

2.4 Resiliency to Sensor-Node Capture

The unattended operation of sensors in hostile areas raises the real possibility of sensor-node capture by an adversary. Although node capture is a general threat that affects all security mechanisms, not just a node's key ring, it is worth examining the resiliency of a key management scheme to such a threat.

We distinguish between two levels of threats posed by node capture and potential countermeasures. The first is that of active manipulation of a sensor's data-inputs. Although this threat does not necessarily require a physical attack against a sensor, it does imply that an adversary can disseminate bogus data in the DSN. Such an attack cannot usually be prevented and it may not be practical, or even possible, to detect it by physical DSN surveillance (e.g., by satellite or aerial imagery). In general, detection of such attacks is especially difficult since sensor nodes may not necessarily communicate in an erratic or anomalous manner. Hence, traditional anomaly-detection techniques may not apply. Detecting a sensor's data input manipulation may require data correlation analysis and data-anomaly detection, possibly off-line, by collection and processing nodes. While such analysis can detect active insertion of bogus data by an adversary, it requires redundant sensor coverage of deployment areas and, hence, sufficient sensor-node density in the DSN.

The second level of threat materializes when a sensor node is under the complete physical control of the adversary. This level includes the first, and in addition enables an adversary to mount attacks against other sensors of the DSN. For example, an adversary can launch a "sleep-deprivation attack" [13] that may exhaust the batteries of the sensor nodes with whom the captured node shares keys by excessive communication. Handling sensor-node capture typically requires that tamper-detection technologies [7, 13, 14] be used to shield sensors in such a way that physical sensor manipulation would cause the erasure of the sensor's key ring and the disabling of the sensor's operation[3]. For some sensor designs, it may be practical to encrypt a node's key ring in a key-encrypting key whose erasure can be very fast.

Although we assume tamper-detection via sensor-node shielding that erases the keys of captured nodes, we note that our key-distribution scheme is more robust than those based on a single mission key or on pair-wise private sharing of keys even in the face of physical attacks against captured unshielded sensor nodes. In the single mission key scheme, *all* communication links are compromised, whereas in the pair-wise private key sharing, all *n-1* links to the captured unshielded node are compromised. In contrast, in our scheme only the $k \ll n$ keys of a single ring are obtained, which means that the attacker has a probability of approximately $\frac{k}{P}$ to attack successfully any DSN link (viz., simulation results of Section 4). The node's shared keys with controllers could also be re-created by the adversary, but this does not affect any other sensor nodes.

3. ANALYSIS

3.1 DSN Connectivity with Random Graphs

The limits of the wireless communication ranges of sensor nodes, not just the security considerations, preclude use of DSNs that are fully connected by shared-key links between all sensor nodes. For example, two nodes that are not in wireless communication range cannot take advantage of their shared key in a fully connected network. Moreover, it is unnecessary for the shared-key discovery phase to guarantee full connectivity for a sensor node with all its neighbors in wireless communication range, as long as multi-link paths of shared keys exist among neighbors that can be used to setup a path key as needed. Further, extra shared-key provisioning is required for incremental network growth and, possibly, for path-key establishment following revocation and re-keying.

Let p be the probability that a shared key exists between two sensor nodes, n be the number of network nodes, and $d = p * (n-1)$ be the expected degree of a node (i.e., the average number of edges connecting that node with its graph neighbors). To establish DSN shared-key connectivity, we need to answer the following two questions:

- what value should the expected degree of a node, d, have so that a DSN of n nodes is connected? and,
- given d and the neighborhood connectivity constraints imposed by wireless communication (e.g., the number of nodes n' in a neighborhood), what values should the key ring size, k, and pool, P, have for a network of size n? In particular, if memory capacity of each sensor limits the key ring size to a given value of k, what should the size of the key pool, P, be?

[3]The problem of detecting and handling sensor-node capture is reminiscent of the somewhat similar concerns regarding the fast destruction of cryptographic keys by British agents captured on enemy territory during World War II. For example, the Special Operations Executive equipped its agents in the occupied Europe with cryptographic keys printed on silk, which could be easily camouflaged in a coat's linings, cut and burnt. Other examples of the fundamentally difficult problem of detecting and handling agent capture are provided by Leo Marks' vivid account [10].

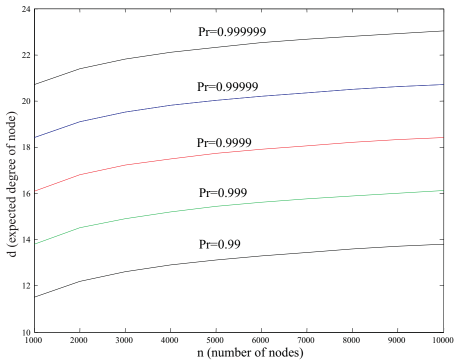

Figure 1: Expected degree of node vs. number of nodes, where $P_c = Pr[G(n,p)$ is connected]

Random-graph theory helps answer the first question. A random graph $G(n,p)$ is a graph of n nodes for which the probability that a link exists between two nodes is p. When p is zero, the graph does not have any edge, whereas when p is one, the graph is fully connected. The first question of interest is what value should p have such that it is "almost certainly true" that the graph $G(n,p)$ is connected.

Erdős and Rényi [12] showed that, for monotone properties, there exists a value of p such that the property moves from "nonexistent" to "certainly true" in a very large random graph. The function defining p is called the threshold function of a property. Given a desired probability P_c for graph connectivity, the threshold function p is defined by:

$$P_c = \lim_{n \to \infty} Pr[\text{G(n,p) is connected}] = e^{e^{-c}}$$

where

$$p = \frac{ln(n)}{n} + \frac{c}{n} \text{ and c is any real constant.}$$

Therefore, given n we can find p and $d = p * (n-1)$ for which the resulting graph is connected with desired probability P_c.

Figure 1 illustrates the plot of the expected degree of a node, d, as a function of the network size, n, for various values of P_c. This figure shows that, to increase the probability that a random graph is connected by one order, the expected degree of a node increases only by 2. Moreover, the curves of this plot are almost flat when n is large, indicating that the size of the network has insignificant impact on the expected degree of a node required to have a connected graph.

To answer the second question above, we note that the wireless connectivity constraints may limit neighborhoods to $n' \ll n$ nodes, which implies that the probability of sharing a key between any two nodes in a neighborhood becomes $p' = \frac{d}{(n'-1)} \gg p$. Hence, we set the probability that two nodes

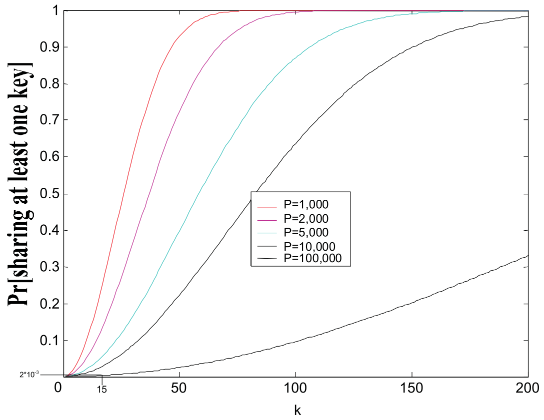

Figure 2: Probability of sharing at least one key when two nodes choose k keys from a pool of size P

share at least one key in their key rings of size k chosen from a given pool of P keys to p' and then derive P as a function of k. This derivation takes into account that the size of the key pool, P, is not a sensor-design constraint. In contrast with k, which is limited by the sensor memory size, the key pool is generated and used off-line and hence its size, P, can be very large. To derive the value of P, given constraint k for a p' that retains DSN connectivity with an expected node degree d, we note that $p' = 1 - Pr$[two nodes do not share any key], and thus

$$p' = 1 - \frac{((P-k)!)^2}{(P-2k)!P!}$$

(viz., derivation in Appendix A[4]). Since P is very large, we use Stirling's approximation for $n!$,

$$n! \approx \sqrt{2\pi} n^{n+\frac{1}{2}} e^{-n}$$

to simplify the expression of p', and obtain:

$$p' = 1 - \frac{(1-\frac{k}{P})^{2(P-k+\frac{1}{2})}}{(1-\frac{2k}{P})^{(P-2k+\frac{1}{2})}}$$

Figure 2 illustrates a plot of this function for various values of P. For example, one may see that for a pool size P = 10,000 keys, only 75 keys need to be distributed to any two nodes to have the probability $p = 0.5$ that they share a key in their key ring. If the pool is ten times larger, namely P = 100,000, the number of keys required is 250, which is only 3.3 times the number of keys distributed in the case P = 10,000. This provides intuition for the scalability of our approach. Of course, to determine the final the size of the key ring we need to provision for addition of new nodes, revocation, and re-keying. The scalability properties of our solution indicate that such provisioning will have minimal impact on the size of key rings.

[4]We could have used the "birthday paradox" to derive the formula for p' but that would have given us only an approximation of p' since the keys of a ring are drawn out of the pool of size P without repacement.

3.2 An example

To understand how the scheme works we present a simple numerical example. Let us assume that a DSN has n=10,000 nodes and that we want the resulting network to be connected with probability $P_c = 0.99999$. This means the network will "almost certainly" be connected. Further, assume that that each node in the DSN has a wireless communication range that requires a neighborhood connectivity of 40 nodes.

Using the Erdős and Rényi's formula we find that $c = 11.5$. For this value of c we obtain $p = 2 * 10^{-3}$ and $d = 2 * 10^{-3} * 9999$. It follows that if in our network each node can communicate with, on the average, 20 other nodes out of the $n = 10,000$ nodes, the network will be (almost certainly) connected. The formula of p' above shows that if we set $p' = p = 2*10^{-3}$ and select an especially small value of k, say $k = 15$, we must have a pool size $P = 100,000$ (also viz., Figure 2). Of course, larger values of k can be accommodated by a pool size $P = 100,000$, as seen below.

The requirement that each neighborhood consists of $n' = 40$ sensor nodes, implies that instead of $p = 2*10^{-3}$ we now have $p' = \frac{d}{n'-1} = \frac{20}{40-1} \approx 0.5$. This means that either the size of the key ring, k, or the pool size, P, or both, must increase. For example, the formula for p' above indicates that we now need to increase the key ring size k from 15 to 250 if we intend to use the same pool size $P = 100,000$. Furthermore, if the neighborhood size is increased to $n' = 60$, then $p' = \frac{20}{60-1} \approx 0.33$. The formula for p' above indicates that we now need only a key ring size of $k = 200$ for a pool size of $P = 100,000$ keys.

4. SIMULATIONS

We use simulation to investigate the effect of the various parameters on different DSN sizes. Of particular interest are the efficiency and scalability of our scheme and also the determination of some parameter values that cannot be easily computed, such as the diameter of the resulting secure network.

The simulations assume a network of 1,000 nodes with an average density of 40 sensor nodes in a neighborhood. Each simulation is run 10 times with different seeds for the random number generator, and the results presented represent the average values on the 10 runs, unless otherwise noted.

4.1 Effect on the network topology

The fact that two nodes may not share a key during the shared-key discovery phase means that, from a network router's point of view, a link does not exist between those two nodes. This has an effect on the average path length (i.e., the number of links) between two nodes after shared-key discovery. We compute this value for various sizes of the key ring and show the result on Figure 3. This figure indicates that the average path length of the network depends on the size of the key ring. The smaller k is the higher the probability that a link does not have a key and, therefore, longer paths need to be found between nodes. In this example, the network gets disconnected for small k.

Because some links may not be keyed, a node may need to use a multi-link path to communicate with one of its wireless neighbors. Although this path would be used only once (to send the key to use for the link encryption), it should not be very long; otherwise the delay and communication cost to setup a path key with a neighbor may be high. In this

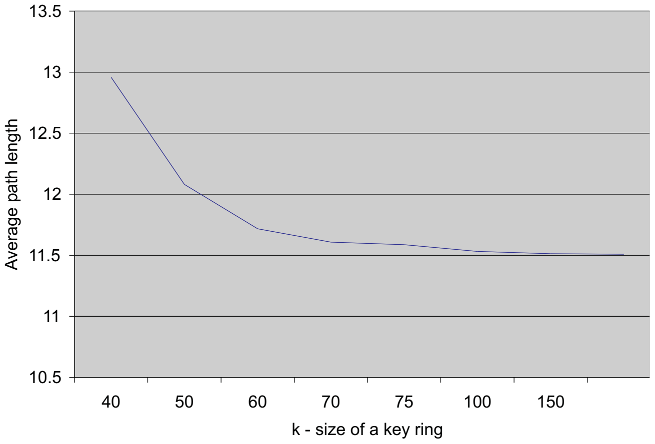

Figure 3: Average path length at the network layer

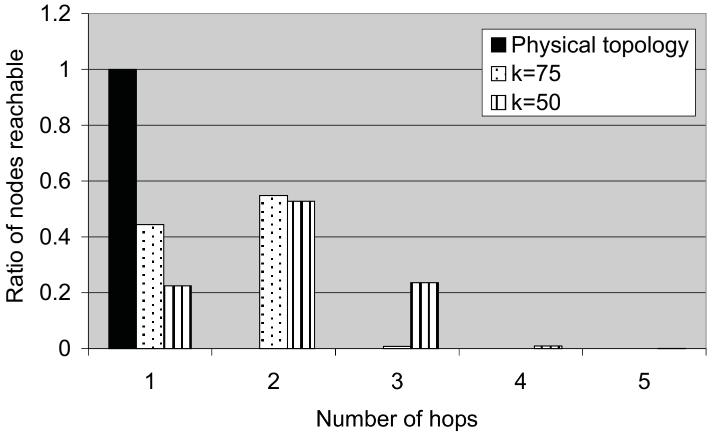

Figure 4: Path length to neighbors

example, we show how the multi-link path from a node to one of its neighbor varies with k.

Figure 4 shows that the effect of traversing multiple links (hops) to set up a path key is negligible. If a neighborhood node cannot be reached via a shared key (i.e., one link or one hop), it will take at most two or three links to contact it. Since this has to be done only once to setup the path key, the effects are negligible. With $k = 75$, only half of the neighbors are reachable over a single link, but most of the other may be reachable over in three-link paths. While for $k = 50$ only one third of the nodes are reachable over a single link, but at most four links are needed for a path to contact all of them.

4.2 Effect of an Attack against Unshielded Sensor Nodes

We suggested that capture of an unshielded node leads to the compromise of only k keys and that an adversary could only attack $\frac{k * \text{number of links}}{P}$ links. We verified this fact by observing how many keys are used to secure links in the simulated DSN and how many links are secured with the same key.

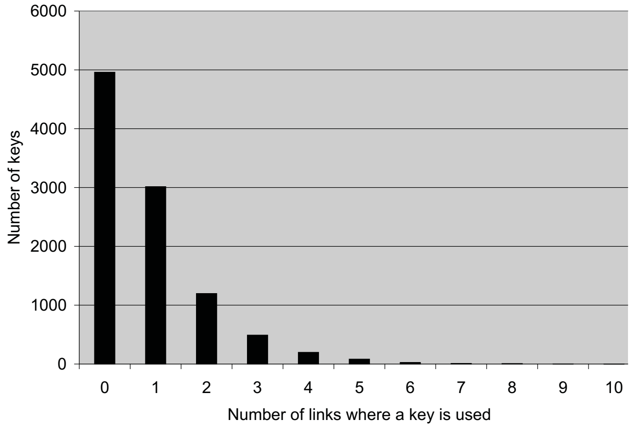

Figure 5: Usage of the key pool (P = 10,000)

Figure 5 shows that, out of the pool of 10,000 keys, only 50% of the keys are used to secure links, only 30% are used to secure one link, 10% are used to secure two links, and only 5% are used to secure 3 links. This suggests that compromise of one key does lead to the compromise of another link with probability 0.3, of two other links with probability 0.1, and so on.

5. CONCLUSIONS

We presented a new key management scheme for large-scale DSNs. All such schemes must be extremely simple given the sensor-node computation and communication limitations. Our approach is also scalable and flexible: trade-offs can be made between sensor-memory cost and connectivity, and design parameters can be adapted to fit the operational requirements of a particular environment. We illustrated the effect of modifying design parameters using both analysis and simulations. The results indicate that our scheme is superior to the traditional key pre-distribution schemes.

Acknowledgements

We would like to thank Rakesh Bobba, Himanshu Khurana, and Radostina Koleva for helpful comments and discussions. We also thank the anonymous reviewers for many valuable comments.

Disclaimer

The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, the Army Research Office, or the U.S. Government.

6. REFERENCES

[1] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung, "Perfectly Secure Key Distribution for Dynamic Conferences," in Advances in Cryptology — CRYPTO '92, LNCS 740, Springer-Verlag, Berlin, August 1993, pp. 471–486.

[2] C. Blundo, L. A. Frota Mattos and D. R. Stinson, "Tradeoffs Between Communication and Storage in Unconditionally Secure Schemes for Broadcast Encryption and Interactive Key Distribution," Advances in Cryptology – CRYPTO '96, LNCS 1109, Springer Verlag, Berlin, August 1996, pp. 387–400.

[3] D. W. Carman, P. S. Kruus and B. J. Matt,"Constraints and Approaches for Distributed Sensor Network Security," dated September 1, 2000. NAI Labs Technical Report #00-010, available at *http://download.nai.com /products/media/nai/zip/nailabs-report-00-010-final.zip*

[4] A. Fiat and M. Naor, "Broadcast Encryption," in Advances in Cryptology — CRYPTO '93, LNCS 773, Springer-Verlag, Berlin, August 1993, pp. 480–491.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, "System architecture directions for network sensors," Proc. of ASPLOS-IX, Cambridge, Mass. 2000.

[6] V.D. Gligor and P. Donescu, "Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes," Fast Software Encryption 2001, M.Matsui (ed), LNCS 2355, Springer Verlag, April 2001.

[7] IBM, IBM 4758 General Information Manual, available at *http://www.ibm.com/security/cryptocards/*

[8] C.S. Jutla, "Encryption Modes with Almost Free Message Integrity," Advances in Cryptology - EUROCRYPT 2001, B. Pfitzmann (ed.), LNCS 2045, Springer Verlag, May 2001.

[9] J. M. Kahn, R. H. Katz and K. S. J. Pister, "Mobile Networking for Smart Dust," ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99), Seattle, WA, August 17-19, 1999, pp. 271 - 278.

[10] Leo Marks, *Between Silk and Cyanide - A Codemaker's War, 1941-1945*, A Touchstone Book, Simon & Schuster, Inc., 2000.

[11] P. Rogaway, M. Bellare, J. Black, and T. Krovetz,"OCB: A Block-Cipher Mode of Operations for Efficient Authenticated Encryption," Proc. of the 8th ACM Conf. on Computer and Communication Security, Philadelphia, Penn., November 2001.

[12] J. Spencer, *The Strange Logic of Random Graphs*, Algorithms and Combinatorics 22, Springer-Verlag 2000, ISBN 3-540-41654-4.

[13] F. Stajano, *Security for Ubiquitous Computing,* John Wiley and Sons, New York, Feb. 12, 2002, ISBN: 0-470-84493-0, 267 pp.

[14] S.R. White and L. Comerford, "ABYSS: An Architecture for Software Protection," *IEEE Transactions on Software Engineering,* vol. 16, No. 6, June 1990, pp. 619-629.

Appendix A

The probability that two key rings share at least a key is 1 - Pr[two nodes do not share any key]. To compute the probability that two key rings do not share any key, we note that each key of a key ring is drawn out of a pool of P keys *without* replacement. Thus, the number of possible key rings is:

$$\frac{P!}{k!(P-k)!}$$

Pick the first key ring. The total number of possible key rings that do not share a key with this key ring is the number of key-rings that can be drawn out of the remaining $P-k$ unused key in the pool, namely:

$$\frac{(P-k)!}{k!(P-2k)!}$$

Therefore, the probability that no key is shared between the two rings is the ratio of the number of rings without a match by the total number of rings. Thus, the probability that there is at least a shared key between two key rings is:

$$\frac{k!(P-k)!(P-k)!}{P!k!(P-2k)!}$$