Analysis and Evaluation of SECOS, a Protocol for Energy Efficient and Secure Communication in Sensor Networks

Issa Khalil, Saurabh Bagchi, Ness Shroff[1],
Dependable Computing Systems Lab
School of Electrical and Computer Engineering
Purdue University.
Emails: {ikhalil, sbagchi, shroff}@purdue.edu

Contact Info for Corresponding Author: 465 Northwestern Avenue, West Lafayette, Indiana 47907. USA.

Phone: 765-494-3362 Fax: 765-494-2706

Abstract

Wireless sensor networks are increasingly being used in applications where the communication between nodes needs to be protected from eavesdropping and tampering. Such protection is typically provided using techniques from symmetric key cryptography. The protocols in this domain suffer from one or more of the following problems — weak security guarantees if some nodes are compromised, lack of scalability, high energy overhead for key management, and increased end-to-end data latency. In this paper, we propose a protocol called SECOS that mitigates these problems in static sensor networks. SECOS divides the sensor field into control groups each with a control node. Data exchange between nodes within a control group happens through the mediation of the control head which provides the common key. The keys are refreshed periodically and the control nodes are changed periodically to enhance security. SECOS enhances the survivability of the network by handling compromise and failures of control nodes. It provides the guarantee that the communication between any two sensor nodes remains secure despite the compromise of any number of other nodes in the network. The experiments based on a simulation model show a seven time reduction in energy overhead and a 50% reduction in latency compared to SPINS, which is one of the state-of-the-art protocols for key management in sensor networks.

Keywords: sensor network security, key management, symmetric cryptography, energy efficient key distribution, key refreshment.

[1] Corresponding author

1 Introduction

Sensor networks are being deployed in situations where it is important to protect the message communication from eavesdropping or tampering. The deployments in military situations in hostile territory have strict security requirements for message communication. Some deployments in civilian situations have security requirements as well. Consider a patient monitoring system that uses biological sensors placed *in situ* in the patient. The communication should be secured for privacy reasons. A sensor network used for monitoring environmental conditions in public places (such as, concentration of toxins in the air, biometric sensors in airports) should have its inter-node communication protected against tampering as a guard against possible terrorist attacks directed to critical civilian infrastructures. These networks must also continue to function correctly in the event of certain nodes being taken over by an adversary.

Cryptography is the foundational technology used for protecting and securing the communication in sensor networks. This technology relies on keys as the centerpieces, and many attacks focus on disclosing these keys. This makes the management of the keys (the process by which keys are generated, stored, protected, distributed, used, and destroyed) in a large-scale network of up to hundreds of thousands of sensor nodes a very important and challenging problem. Sensor nodes are constrained in their energy availability, memory and computational resources, and communication bandwidth. These constraints make it impractical to use *asymmetric* algorithms for key management. These algorithms are very computationally intensive, and consequently, energy intensive since at their heart they involve exponentiation and modulus operations of large numbers. The common approach, therefore, is to use symmetric key cryptography where the two end-points of a communication share a secret key. The challenge is to manage the keys for symmetric cryptography in a *scalable* manner. The scalability goal implies that the end-to-end communication delay, energy overhead for key management, and the dollar cost of deployment should increase gradually with increasing size of the sensor network. Since the sensor nodes may be placed in hostile environments, we must also design for the possibility that some nodes may be taken over or compromised. The sensor nodes are inherently less reliable than wired platforms and therefore, a protocol must be designed to function in the face of some nodes being unavailable. Radio communication is recognized as more energy consuming than computation by several orders of magnitude[48]. Consequently, the key management protocol should minimize the number of overhead control messages and the overhead number of bytes added to data messages.

Some symmetric key management protocols rely on a common shared secret key between all the nodes in the network leading to a highly insecure deployment. At the other end of the spectrum, some protocols have a separate shared key for each pair of nodes, which leads to a large amount of key storage that grows as the square of the number of nodes, and is therefore not scalable. The requirement to minimize communication overhead makes most of the proposed purely symmetric algorithms impractical since they add a fixed size overhead number of bytes to the payload and sensor networks typically have small sized packets.

In this paper, we propose and analyze a protocol called SECOS (**S**calable & **E**nergy-Efficient Secure **C**ommunication **O**n **S**ensors) for key management in static sensor networks that uses symmetric cryptography. Our high-level design goals in SECOS are to (i) provide a scalable and secure key distribution channel for any-to-any communication in a large-scale sensor network, (ii) minimize the adverse fallout of compromising any sensor node, (iii) make key management energy efficient, and (iv) reduce the end-to-end delay of secure data communication.

Using the well-known approach of node clustering [41]-[44], SECOS divides the sensor field into multiple control groups and assigns a rotating control node to each group. Communication within a group occurs through the use of keys exchanged with the help of the control node, while inter-group communication involves establishing a secure channel between the respective control nodes through the involvement of the base station. Effectively, SECOS imposes a three-level hierarchy of the nodes – a single base station, multiple control nodes, and a large number of sensing nodes. Of these, only the base station is fixed, assumed to be secure and assumed not to have any resource constraints, while all the rest, including the control nodes, are generic sensor nodes. Although node clustering is a well-known technique, it has to be used with special care for key distribution to protect the network against the compromised nodes that play a special role in node clustering. The control nodes are assumed to be susceptible to compromise and are monitored and can be removed from their privileged role. SECOS also provides techniques for secure initial deployment and revocation of suspect nodes.

A key decision choice in SECOS is the control group size. We present a simple mathematical analysis to determine an upper bound on the control group size, due to the resource constraints on the control node and the allowable security. We then present an equation that quantifies the energy cost of key management in terms of several factors, including the control group size, and derive the optimal control group size for the most energy-efficient key management.

A promising approach for sensor key management has been proposed in a system called SPINS [1]. SPINS uses the base station as an intermediary for secure communication between any two nodes. We create a simulation model for comparing SECOS and SPINS with respect to end-to-end data latency and energy overhead of key management. For a fair comparison, we make the key caches also available to SPINS, though the original work does not mention caches. The simulation results

show that SECOS reduces the energy consumption by a factor ranging from 1.2 to 7 and the end-to-end data latency by a factor of 1.05 to 1.50 depending on the communication pattern and the cache size. A large cache means keys are available locally and then SECOS performs comparably to SPINS. However, this also implies additional storage requirement and the deployment is less secure to nodes being compromised. We provide a mathematical analysis to quantify the probability of exposing the communication between two legitimate nodes as a function of the number of compromised nodes. This is done for SECOS, SPINS [1], and a key pre-distribution protocol due to Du [19] and SECOS is shown to perform better for large operating regions.

Many key management protocols for ad-hoc networks have been proposed in the literature. They suffer from one or more of the problems of weak security guarantees if some nodes are compromised, lack of scalability, high energy overhead for key management, and increased end-to-end data latency. In general, the key pre-distribution protocols [2],[11],[15]-[19],[21]-[24],[29] expose the security of the whole network when a certain fraction of nodes is compromised. Kerberos-like protocols (such as, [1]) divide the network into several sections with privileged nodes for key management in each section. If the privileged node fails or is compromised, secure communication in the entire section becomes impossible. A detailed comparison with existing schemes is presented in Section 6.

Our paper makes the following contributions.

1. It provides a scalable protocol for key management that is sensitive to the sensor node's resource constraints, including computation, communication, and bandwidth. We believe that current technology trends may remove some of the resource constraints, such as memory and processing power, in the foreseeable future, while the constraints of bandwidth and energy are expected to remain for some time to come.
2. It presents an energy efficient method for key management and substantial energy savings are demonstrated without introducing specialized high cost nodes in the network.
3. The protocol is resilient to some nodes being compromised due to attacks. In fact, it guarantees that, under a given set of assumptions, the communication between two uncompromised nodes cannot be exposed, irrespective of the number of other nodes that are compromised. Similarly, the protocol can tolerate some nodes being unavailable due to natural failures.

SECOS uses several techniques well-known in the network security domain, such as node clustering, key refreshment, and neighbor watch. Its contribution lies in synthesizing the different techniques into a cohesive protocol and applying that to the sensor network environment, with its distinctive constraints, chiefly, energy and susceptibility of the nodes to being physically compromised. We show that SECOS performs better with respect to existing state-of-the-art protocols for large parts of the normal operating region of sensor networks. In this paper, we *do not* describe the design in SECOS to address all forms of ID spoofing attacks and secure node addition to the existing network.

The rest of the paper is organized as follows. Section 2 presents the design of SECOS. Section 3 discusses how SECOS handles different classes of attacks. Section 4 presents a mathematical analysis for the maximum control group size and the energy-wise optimal control group size. Section 5 describes the experiments and the results. Section 6 refers to related research. Section 7 concludes the paper.

2 Description of SECOS

We use a few basic well-known techniques in the design of SECOS.

1. *Refreshing the keys and purging the caches*. The keys are periodically refreshed and the key caches are purged regularly for two important security goals. The first is to minimize the adverse fallout of compromising some nodes in terms of the number of *old* messages that are exposed. The second goal is to defeat possible cryptanalysis attacks by analyzing plaintext and ciphertext pairs processed with the same encryption key.
2. *Changing the nodes which play a privileged role*. We do not wish to assume a large number of specialized well-protected nodes in our environment. Therefore, we design for the possibility of the nodes with special key management functionality being compromised and provide for them to be changed either on a time schedule, or when triggered by anomalous events. Another important goal of the control role rotation among the members of the control group is to achieve load balancing and even energy drain since the control node's activities are more demanding.
3. *Neighbor watch*. Each node maintains a list of its immediate neighbors and can overhear neighborhood traffic in order to detect compromised nodes.

2.1 System Assumptions and Attack Model

System assumptions: We assume that the links are bi-directional, which means that if a node *A* can hear node *B* then *B* can hear *A*. Also, we assume that the network has a static topology, though the functional roles a node plays (e.g., cluster head, data aggregator, etc.) may change. Also we assume that the sensor nodes are distributed uniformly on the sensor field.
We assume that the base station in SECOS is secure, not prone to failures, and does not have any resource constraints (bandwidth, energy, etc.). Protection against failures can be achieved by fault tolerant techniques such as redundancy for

natural failures, or through a variety of possibly expensive security mechanisms, such as tamper proof hardware, for malicious failures. We assume that there is a certain amount of time from a node's deployment, called the *compromise threshold time* (T_{Comp}), that is minimally required to compromise the node. We believe as in [24], [49], [50], that a sensor node deployed in a security critical environment must be designed to sustain possible break-in attacks at least for a short interval (say several seconds) when captured by the adversary; otherwise, the adversary could easily compromise all the nodes and thus take over the network. Therefore, instead of assuming that sensor nodes are tamper resistant which often turns out not to be true and very expensive, we assume there exists a lower bound on the time interval T_{comp} that is necessary for an adversary to compromise a sensor node, and that the time T_{ND} for a newly deployed sensor node to discover its immediate neighbors is smaller than T_{comp}. In practice, we expect T_{ND} to be of the order of several seconds, so we believe it is a reasonable assumption that $T_{comp} > T_{ND}$. The current generation of sensor nodes can transmit at the rate of 40 Kbps [51] whereas the size of an ID announcement message is very small (12 bytes if an ID is 4 bytes and the hardware address size is 8 bytes).The probability of collision is quite small when a non-persistent CSMA protocol is used for medium access control [52]. Moreover, a node can broadcast its ID multiple times to increase the probability that it is received by all its neighbors. Furthermore, we assume that no external node exists in the network during the neighbor discovery.

Attack model: A malicious node can be either an external node that does not know the cryptographic keys, or an insider node, that possesses the keys. An insider node may be created, for example, by compromising a legitimate node. All these malicious nodes can exhibit Byzantine behavior and can collude amongst themselves. Any malicious node can for example eavesdrop on the traffic, inject new messages, replay and change old messages, spoof other identities, or pass traffic from one location of the network to a colluding node in another location (wormhole attack).

2.2 Keys in SECOS

SECOS uses five types of keys: the master key, the volatile secret key, the session key, the authentication key (*MAC* key), and the pseudo random number generator key (seed).

Some notations. We will use the following notations for keys in the paper. K_{AB} ($=K_{BA}$) refers to any secret key shared between A and B. The five kinds of keys – the master key, the volatile secret key, the session key, the Authentication (*MAC*) key, and the random number generator key, will be denoted respectively as MK_{AB}, VK_{AB}, SK_{AB}, AK_{AB}, and RK_{AB}. $E(K,X)$ denotes the encryption of a message X using key K. $MAC(K,Z \oplus X||Y)$ refers to the application of the *MAC* algorithm, keyed by key K, to the result of the concatenation of Y with the result of Z xor-ed with X. $H(X)$ is the hash value of X. Any symmetric key encryption algorithm suitable for sensor networks may be used for encryption and decryption. It is desirable that the cipher text be the same length as the plaintext in order to reduce the message transmission overhead. An example of such a protocol is the counter mode (CTR) of block ciphers [14],[16]. Any underlying block cipher algorithm could be used with the CTR mode, e.g. DES [36] and its variants 3DES and DES-X, Rijindael [37], AES [37], TEA [38], and RC5 [39].

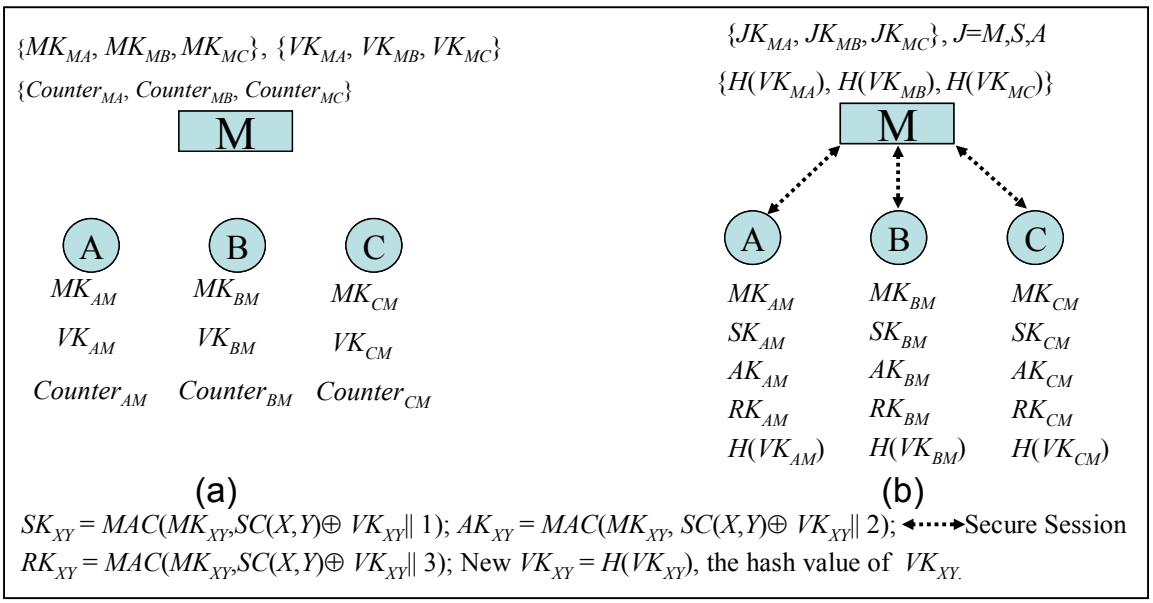

Figure 1: Initial key setup between base station and three sensing nodes

The master key is burnt into each sensor node at manufacture time and is shared with the base station. It is not used for encrypting message communication channels, but instead to generate other keys to be used for encryption and authentication. Compromising the communication channel does not reveal the master key since it is not used in any channel communication. The volatile secret key is also shared between the node and the base station. It is used, along with the

master key, to generate the session and *MAC* keys. After each generation of session and *MAC* keys, a new volatile secret key is generated by applying a hash function to the current volatile secret key, after which the current one is deleted and replaced by the new one. This provides SECOS with forward secrecy; if a node gets compromised, previous communications of the node are not exposed. This is due to the fact that the attacker is not able to generate the old keys since the earlier volatile secret keys are not available at the time of compromise, even though the master key is. As in the case of the master key, crypt-analyzing the communication does not reveal the volatile secret key since it is not used in any channel communication.

The base station also shares two counters with each sensing node, one for each direction (sending and receiving) of communication *SC(M,S)* and *RC(M,S)*. These counters are kept synchronized by incrementing them on messages sent or received between the sensor and the base station. During synchronization, the receive-counter value at one party is matched with the send-counter value at the other party. However, the counters need not to be exactly synchronized; they can be off by some known number *Sync_diff*. When the counters are not synchronized, the key generated at the base station using *SC(M,S)* may not match the one generated at the sensor node using *RC(S,M)*. Therefore, the sensor node adjusts (increments/decrements) *RC(S,M)*, generates the key, and compares the key with that generated by *M*. The sensor node continues to do that until the keys are either matched or the number of adjustments to *RC(S,M)* equals *Sync_diff*. In the latter case the sensor nodes initiate counter synchronization with the base station. In addition to the conventional use of counters to achieve semantic security, they are used in SECOS as a variable input for key generation. The semantic security prevents a malicious node from replaying old, properly authenticated messages that was used to establish keys between legitimate nodes. The use as the variable input is required in the key generation process to introduce randomness. These counters are used to replace the job of a nonce or a sequence number that ordinarily would be attached to every message to prevent the replay of old messages. However, due to the fact that communication is far more energy consuming than computation [58], we use the shared synchronized counters to minimize the transmission overhead of the sequence number or the nonce with every message. Figure 2 presents an algorithm that is used to synchronize the counters during key refreshment. Therefore, for most of the time, the counter synchronization does not incur any overhead and comes as a by-product of key refreshment. For example during the course of our simulations no counter synchronization is required beyond that with the key refreshment. New keys are generated by applying *MAC* and hash functions over data that includes these counters. Figure 1(a) shows the initial keying material that includes the master key, the volatile secret key, and the counters.

1. M generates a new session key: $SK_{MS} = MAC(MK_{MS}, SC(M,S) \leftarrow VK_{MS} \| 1)$.
2. M generates a new Authentication key: $AK_{MS} = MAC(MK_{MS}, SC(M,S) \leftarrow VK_{MS} \| 2)$.
3. M ➔ S: $Counter_{MS}$, *Change*, *MAC* (AK_{MS}, $Counter_{MS} \| Change$).
4. S generates a new session key: new (SK_{SM}) = $MAC(MK_{SM}, RC(S,M) \leftarrow VK_{SM} \| 1)$.
5. S generates a new Authentication key: new (AK_{SM}) = $MAC(MK_{SM}, RC(S,M) \leftarrow VK_{SM} \| 2)$.
6. S generates the next volatile secret key: $VK_{SM} = H(VK_{SM})$.
7. S ➔ M: $Counter_{SM}$, $MAC(AK_{SM}, Counter_{SM})$.
8. M generates the new volatile secret key: new (VK_{MS}) = $H(VK_{MS})$.
9. After the key refreshment is completed, all the old keys are purged.

Figure 2: Key Refreshment and Counter Synchronization Procedure

The rest of the keys are derived from the previous two keys with the help of *MAC* (e.g. HMAC) and hash (e.g. MD5) functions that are preloaded on the base station and the sensors. The session key between the base station and a sensor node is generated by the base station, by applying a *MAC* function over the result of concatenating the binary representation of the number 1 with the result of the *SC(M,S)* XOR-ed with the volatile secret key. The same session key is generated by the sensor node by applying a *MAC* function over the result of concatenating the binary representation of the number 1 with the result of the *RC(S,M)* XOR-ed with the volatile secret key. The *MAC* function is keyed by the master key as shown in the bottom of Figure 1 for SK_{XY}. The purpose of the session key is to provide data confidentiality for communication between two nodes. A similar mechanism is used to generate a shared authentication key between the base station and the sensing node with concatenation of the binary representation of the number 2 instead of the number 1, as shown in the bottom of Figure 1 for AK_{XY}. SECOS uses independent keys for encryption and authentication since it prevents any potential interaction between the primitives that might introduce a weakness and is therefore a good security design principle. SECOS uses the standard key refreshment procedure for the session key and the authentication key. The session key and the authentication key are refreshed periodically or when triggered by a certain event, such as the detection of an attack. The pseudo random key is generated by each entity by applying a *MAC* function over the same parameters as for the session key with

concatenation of the binary representation of the number 3. This key is used as a seed for the pseudo random number generator (e.g. RC4), which is used to produce the stream cipher such as in the CTR mode of DES [16]. This key is refreshed only when the pseudo random string it generates is exhausted, which depends on the pseudo random number generator algorithm used.

Sometimes a packet sent from a source may not reach its final destination either due to a malicious event such as a compromised node in the path dropping the packet or due to natural node or link failure. As a result, the shared counters between these two parties may become unsynchronized, and a procedure has to be invoked to resynchronize them. Key refreshment is accompanied by shared counter synchronization between the two parties. However, the counter synchronization could be launched without the need to refresh any key. Figure 2 shows the key refreshment procedure between, the base station, *M*, and a regular sensor node, *S*. The one-bit flag, ***Change***, is used if the counter synchronization is accompanied by key refreshment.

2.3 SECOS Structure

A flat layout with a powerful base station and sensing nodes distributed through the sensor field and the base station being responsible for key management is clearly not scalable to a large number of nodes. This motivates the hierarchical structure of SECOS. The hierarchical structure we propose for SECOS has clusters of sensor nodes based on geographical proximity. Each cluster has a specially designated node called the *Control Node,* which plays a privileged role for key management. The cluster is called a *Control Group*. SECOS does not impose any special requirements on the control node, and it can be any ordinary sensor node in the cluster. This has the advantage of reducing the possibility of targeted DOS attacks to the specialized nodes. The control node acts as the intermediary for key management. It is periodically changed for the purpose of security (the control node may get compromised), and for more even energy drain (the control node and its neighboring relay nodes drain energy faster). This hierarchical structure shown in Figure 3 consists of three levels of nodes. The root is the base station that is assumed to have powerful resources and is well protected. The internal nodes are regular sensor nodes selected to play the role of control nodes. The leaves are regular sensor nodes.

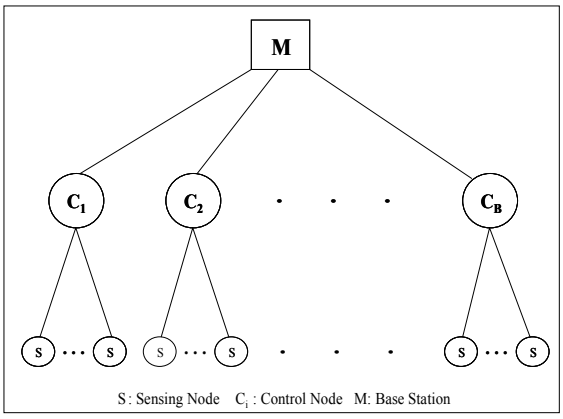

Figure 3: Three level hierarchy for key management in SECOS

An important parameter in SECOS is the size of the control group. The size has two sets of determining factors, which exert opposing effects. The size has to be bounded within a maximum due to three factors — the resource constraints of the control node, such as the communication bandwidth and the computation capacity; the security concerns of not exposing too many nodes if the control node is compromised; and limiting the energy overhead of intra-group key management by bounding the distance between a sensor node and its control node. However, the size has to be kept above a threshold so that most communication occurs within a control group rather than involving multiple control groups since intra-group communication is more energy efficient than inter-group communication. Section 4 provides a detailed mathematical analysis of the control group size.

2.4 Topology Building and Maintenance

It is necessary for the base station to have information about the topology of the network and for each node to have some local topology information. Here, we discuss how such information is initially obtained and subsequently how it is updated and maintained.

As mentioned earlier in Section 2.2, each sensor shares a master key, a volatile secret key, and two counters with the base station from which each sensor node, upon deployment, computes shared session and *MAC* keys with the base station. As a result, a secure session is established between each node in the network and the base station. Also, in the initial deployment phase of the network, each node builds a list of its neighbors and communicates this list to the base station. We assume that a node cannot be compromised and no external malicious nodes exist within the time it takes to build this list,

thus implying that the base station gets a correct view of the neighbor information. We say that two nodes, X and Y, are neighbors if X can hear the transmission of Y. Since we only consider bi-directional links, this implies that Y can also hear the transmission of X. The list of neighbors at each sensor node is built by locally broadcasting a HELLO message, which is a small packet holding the ID of the sender, and then receiving a reply message, which is also a small packet holding the ID of the sender from each node that heard the HELLO message. As soon as the sensor nodes are spread in the sensor field, each node S broadcasts the HELLO message. For each reply received, S adds the sender ID to its neighbor list. Then S sends the full list to the base station authenticated using the authentication key shared between S and M (AK_{MS}). Note that neighbor discovery is secure based on our assumption that no malicious nodes exist in the network during the neighbor discovery. Also note that neighbor discovery incurs a relatively negligible overhead since it is performed only once during the deployment of the network which is assumed to be static. This process is shown in Figure 4. The base station uses these lists to build a connectivity graph that represents the initial network topology and from that the control groups. The connectivity graph is built using an $N \times N$ connectivity matrix that is initialized to 0. For every member i in the neighbor list of S that M receives, M sets the entry (S,i) of the connectivity matrix to 1. Using the connectivity matrix with the knowledge of the limits on the control group size and the maximum number of hops in the control group, the base station generates the control groups. For example, to generate the first control group, M adds node number 1 to the group, then the neighbors of node 1 are added, then the neighbors of each neighbor are added, and so on until the full control group is generated.

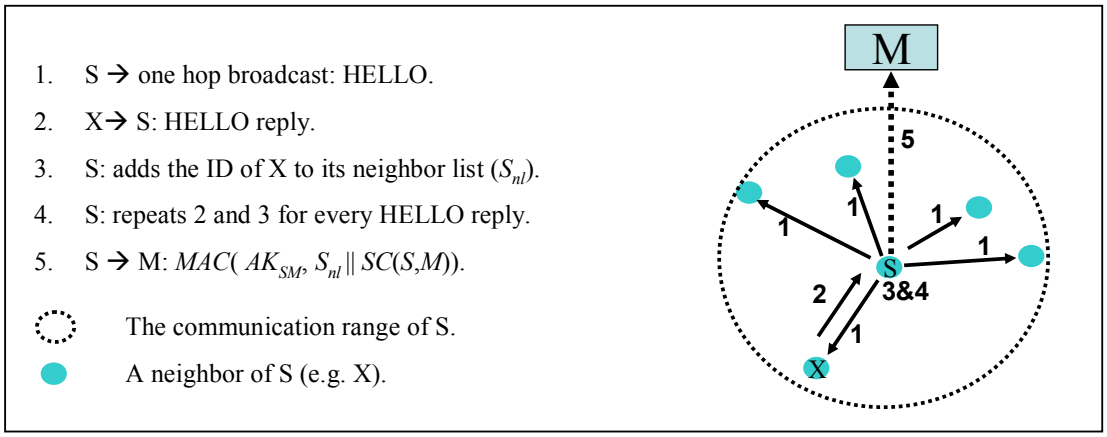

Figure 4 : Building the Topology

Alternately, a secure routing protocol such as INSENS [27] can be used to build the topology information and communicate it to the base station during the routing table construction.

The base station has a global view of the entire network connectivity. When a compromised node is detected, its neighbors are informed, possibly through authenticated multicast [26].

2.5 Assigning and Changing the Control Node

The base station divides the network, based on the topology it built during the setup phase, into control groups consisting of geographically proximal nodes. For each control group, it then designates a node as a control node, say C, and sends it a list of session keys that the base station generates for each node in the group. The list of keys is sent in a message that is encrypted using the shared session key between the control node and the base station (SK_{MC}). The session key is not sent to the sensing nodes in the group. Each sensing node generates that key on its own by applying a *MAC* function over the result of concatenating the binary representation of 1 with the result of the *RC(S,M)* XOR-ed with the volatile secret key shared between the sensor node and the base station. The *MAC* function is keyed by the master key. This process is exactly identical to how the shared session key between the sensor node and the base station is generated independently by both parties as shown in the lower part of Figure 1 for SK_{XY}.

When a sensor node serves as a control node, it does not perform any sensing and uses all its available storage to store the keys. The motivation for this is to restrict the functionality of the control node to key management to facilitate control node monitoring by its neighbors. If the control node were to also send sensory data, it would be impossible for the neighbors to distinguish between control and data traffic since both are encrypted. Also, the key management functionality drains more energy than the regular sensing functionality and we wish to have as even a drain among the different nodes as possible. Finally, the control node requires memory resources to store the keys and does more computations to facilitate key management and we wish to reserve as much resource as possible for the control node to serve its control role. Typically sensor networks have redundant deployments whereby an event can be detected by multiple sensors. This leads us to believe that a reasonable number of nodes (the control nodes) may be exempted from the sensing functionality without adversely affecting the coverage on the sensor field.

After the control node, *C*, receives the list of nodes in the control group, it broadcasts to the group members a message claiming that it is the new control node for the group. This message includes the list of neighbors of the control node that was built during the initial topology discovery phase. When a group member receives the claim, it buffers the claim. When the member needs to use *C*, it challenges *C*. The heart of the challenge lies in generating a random number using the random number generation key introduced earlier, authenticating it with the *MAC* key that should be available at the legitimate control node, asking *C* to do some processing on the number, and send it back authenticated. During this challenge the two nodes establish two shared counters between them. These two counters provide the same functionality as the *SC(M,S)* and *RC(M,S)* that are shared between each node and the base station. If the new control node successfully passes the challenge, the sensor node replaces its current control node with the new one and if it is a neighbor node to the control node, it stores the list of neighbors of the new control node for the purpose of control node monitoring (Section 2.9). Note that now the node has a shared session key with the control node, which is different from the shared session key with the base station. The initial control node set up is shown in Figure 6. Figure 5 shows how a node, *S*, challenge a new control node, say *C*, in addition to the establishment of the shared counters between them.

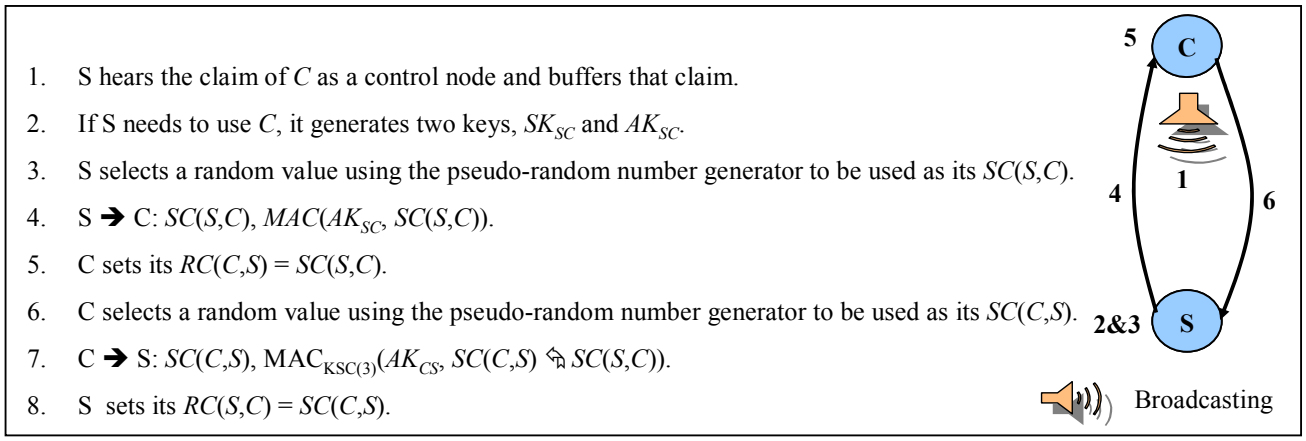

Figure 5: Challenging the Control Node

As mentioned in Section 2, we want to minimize the adverse fall out of a control node being compromised and provide tolerance against control node failures by regularly changing the control node. The control node is changed by the base station based on a certain time schedule, or when some anomalous events are detected, e.g., a compromised control node is detected. When the base station decides to initiate the change, it follows the same procedure as outlined above in this section for a new control node being assigned. In response to the announcement from the new control node, the previous control node, after challenging the new control node and being satisfied, flushes all the cryptographic data in its cache and returns to its normal sensing mode.

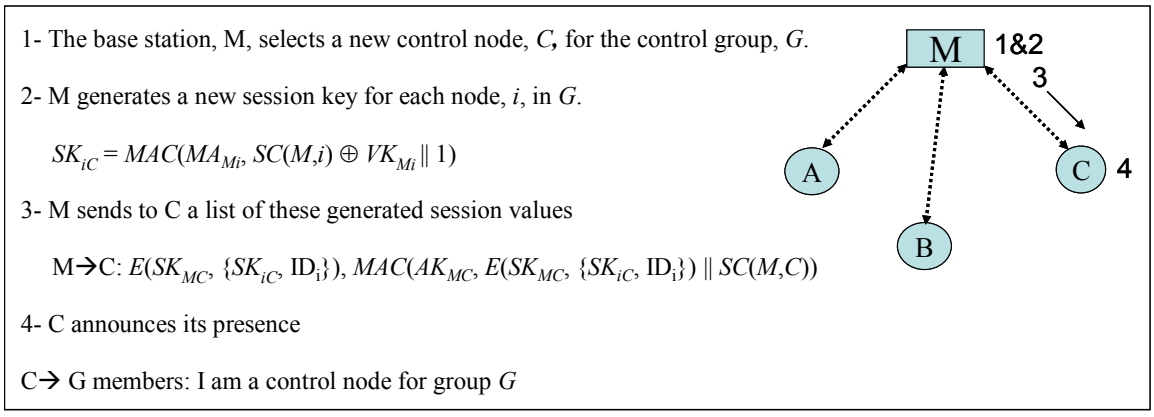

Figure 6: Control node refreshment

2.6 Key Caches

Each sensor node has two types of caches: (i) *Regular cache:* stores the session keys used to encrypt data in message communication between itself and any other node. (ii) *Key request cache:* When a node initiates a data exchange and it does not have the session key for the receiver, it initiates a key establishment process. Subsequently, it may generate more data packets for the same receiver, before the key has been established. The key request cache stores the IDs of such receivers.

In addition, a control node has two types of cache: (i) *Ring cache:* It stores the session keys between itself and each node in its control group. (ii) *Control cache:* It stores the session keys with other control nodes, which are used for inter-group communication.

2.7 Node to Node Communication within Control Group

When a node, say A, needs to communicate with another node within its control group, say B, it first checks in its regular cache for the session key. If present, it uses the cached key. If not present, A generates two random keys K and $\tilde{K}$ and encrypts one of them ($\tilde{K}$) using the other (K) as a key. Let us call K the *Envelop*. Node A sends the encrypted message $E(K, \tilde{K})$ to B. Node A encrypts the key (K) and sends it to the control node C as $E(SK_{AC},K)$. The control node recovers the key K, encrypts it $E(SK_{BC},K)$, and forwards it to the destination B. When B receives the key K from the control node, it can decrypt and obtain the key $\tilde{K}$ that will be used as the shared session key between A and B. When B receives the message that A sent, it stores the message temporarily for the key to arrive from the control node. If B does not receive the key from the control node within a specified time, it drops the packet. Nodes A and B store the session key in their regular cache and continue to use it till the control node is changed, or the key is evicted due to cache replacement. The intra-group communication is shown schematically in Figure 7(a), and the detailed message communication is shown below:

1- $A \rightarrow B$: $A, B, E(K, \tilde{K})$

2- $A \rightarrow C$: $A, B, E(SK_{AC},K), H(K), MAC(AK_{AC}, A \| B \| E(SK_{AC},K) \| H(K) \| SC(A,C))$.

3- $C \rightarrow B$: $A, B, E(SK_{BC},K), H(K), MAC(AK_{BC}, A \| B \| E(SK_{BC},K) \| H(K) \| SC(C,B))$.

The *MAC* function is taken over the encrypted value of the *Envelop*. This has the advantage that the receiver doesn't have to decrypt the *Envelop* if the *MAC* authentication fails, which saves some computation.

2.8 Node to Node Communication across Control Groups

If node A wishes to communicate with a node that lies in a different control group, then two control nodes are involved. Say A lies in group $G1$ and B in $G2$ and the respective control nodes are $C1$ and $C2$. If A does not have the session key with B cached, A generates two random keys (K and $\tilde{K}$) and sends the encrypted message $E(K, \tilde{K})$ directly to B. Node A encrypts the key (K) and sends it to $C1$ as $E(SK_{AC1}, K)$. Node $C1$ checks its control cache for the session key between itself and $C2$. If not present, $C1$ generates a key, say U, and sends it encrypted to the base station as $E(SK_{C1M}, U)$. The base station forwards the key encrypted to $C2$ as $E(S_{MC2}, U)$. Notice that there is no need to send a direct packet from the source control node to the destination control node as in the communication between two nodes within a control group, since the base station is assumed to be trusted. After the session key between $C1$ and $C2$ is established ($SK_{C1C2} = U$), $C1$ sends the key K to $C2$ as $E(SK_{C1C2}, K)$, and $C2$ forwards the key to B as $E(SK_{C2B}, K)$. Node B now has the key K and the message $E(K, \tilde{K})$ from A and proceeds as in the intra-group communication to extract $\tilde{K}$ and use it as the session key.

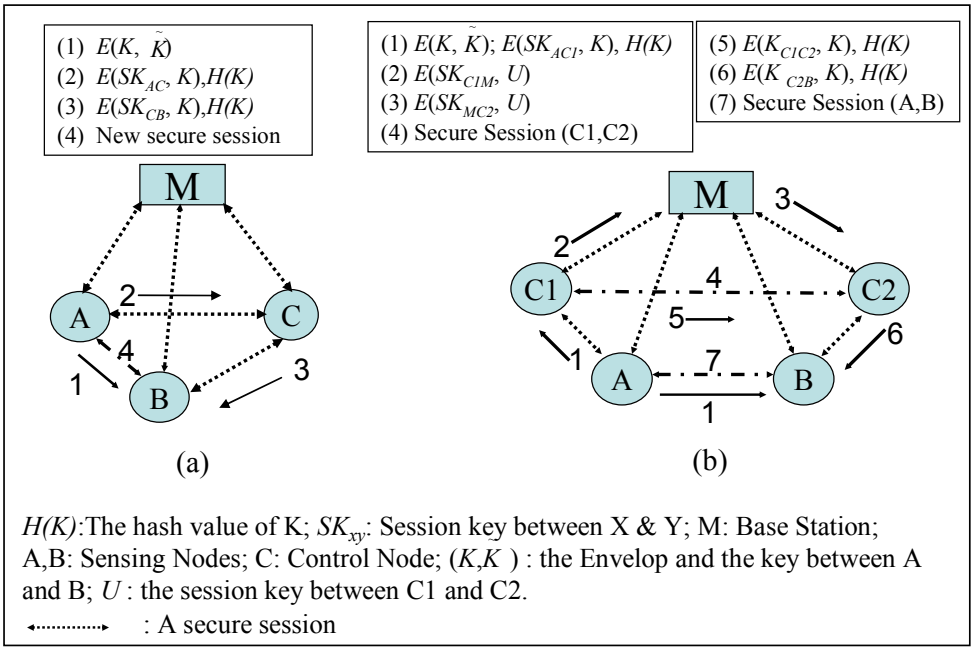

Figure 7: (a) Intra-group communication; (b) Inter-group communication using two control nodes. The two control nodes do not have a secure session when the process starts.

The inter-group communication is shown schematically in Figure 7(b), and the detailed message exchange is shown in the following steps:

1- $A \rightarrow B$: A, B, $E(K, \widetilde{K})$.

2- $A \rightarrow C1$: A, B, $E(SK_{AC1}, K)$, $H(K)$, $MAC(AK_{AC1}, A \| B \| E(SK_{AC1}, K) \| H(K) \| SC(A,C1))$.

3- $C1$ checks its control cache for $C2$, if an entry exists go to step 6.

4- $C_1 \rightarrow M$: C_1, $C2$, $E(SK_{C1M}, U)$, $MAC(AK_{C1M}, C_1 \| C2 \| E(SK_{C1M}, U) \| SC(C1,M))$.

5- $M \rightarrow C_2$: C_1, $C2$, $E(S_{MC2}, U)$, $MAC(AK_{MC2}, C_1 \| C2 \| E(S_{MC2}, U) \| SC(M,C2))$.

6- $C_1 \rightarrow C_2$: A, B, $E(SK_{C1C2}, K)$, $H(K)$, $MAC(AK_{C1C2}, A \| B \| E(SK_{C1C2}, K) \| H(K) \| SC(C1,C2))$.

7- $C_2 \rightarrow B$: A, B, $E(SK_{C2B}, K)$, $H(K)$, $MAC(AK_{C2B}, A \| B \| E(SK_{C2B}, K) \| H(K) \| SC(C2,B))$.

2.9 Monitoring Neighbor Nodes and the Control Node

The control node plays a privileged role in key management and a compromised control node can affect the energy overhead of the network. If the selected control node happens to be compromised, it can launch a DoS attack by refusing to exchange key material among the nodes in its control group. This causes the nodes in the control group to invoke the base station, which fulfils the key request; however this increases the energy consumption since the average number of hops to the base station is higher than that to the control node. Therefore, if the number of key management requests from the same control group goes beyond a threshold, the base station infers that the current control node is misbehaving and assigns a different control node. Hence if the sensor nodes can help the base station choose a probable good node as a control node, then the need to rotate the control nodes prematurely and the number of direct key exchange requests to the base station are reduced. Therefore, SECOS gives each sensor node the option of performing a *neighbor watch*, whereby it observes the source field of the packets going in and out of a neighbor control node. The neighbor watch may be performed by a node at random on a fraction of the packets going in and out of a neighbor control node or with a random periodicity. This fraction or periodicity is determined by the resources and the load at the node. Watching the control node helps in verifying that the control node's behavior does not deviate drastically from the expected functionality for key management. Occasional deviation is expected due to naturally occurring failures. However, one disadvantage of the neighbor watch is blackmailing in which a malicious node falsely accuses a good control node. Therefore, the monitoring is performed cooperatively by all the neighbors of the control node and the nodes in the control group. Our work in [56], [57] presents energy efficient schemes for neighbor watch in sensor networks. Moreover, note that SECOS exchanged keys are secure even if the control node itself is compromised as will be shown in Section 3.1.

Local monitoring is an extension to the *watchdog* [46] concept, which was used to negate the effect on throughput of misbehaving nodes that agree to forward packets but do not. Local monitoring helps detect ID spoofing and Sybil attacks in which an attacker presents one (ID spoofing) or more (Sybil attack [54],[55]) spoofed identities to the network. These identities could either be new fabricated identities or stolen identities from legitimate nodes. Our detailed protocol called DICAS is described in [57], however we provide here a sketch of the detection mechanism. If a malicious node X masquerades as one of its neighbors Y, then the neighbor watch by Y detects this. However, if X masquerades as non-neighbor nodes O, then all the neighbors of X who are not neighbors of O detect the attack since each node knows its neighbors. For example, in Figure 8, if the malicious node X tries to impersonate the non-neighbor node O, then all the neighbors of X, i.e., C, D, Q, P, and Y, will overhear the packet and D, P, and Q, which do not have O in their neighbor lists detect the masquerade and reject the packet.

An opinion about the control node is formed by observing its behavior in response to invocations of its key management routines. Initially, when a node C is assigned the role of a control node, it broadcasts a list of its neighbors. Each neighbor of the new control node sends this list to the base station and also compares the list with its own list of neighbors and marks the common nodes. The base station checks if the control node announced the right list, using its knowledge of the connectivity graph. Figure 8 shows the list of neighbors of the control node C (O, P, Q, X, Y, Z) and some other nodes in the sensor field.

An 8-bit malicious counter (*MalC)* is used to quantify an observer's opinion of a node, with a higher value indicating greater suspicion. After the initial phase when C is assigned as a control node, each neighbor α of the control node starts the monitoring phase by setting $MalC(\alpha,C)$ to zero. A node α is called the *guard node* of a node of C over the link from μ to C if (i) α is a neighbor of C and (ii) μ is a neighbor of both C and α. A guard of C over the link from μ to C monitors the response of C to the key exchange traffic going over that link. For example, in Figure 8, nodes P and Q are neighbors of X as well as C, therefore, they are the guard nodes of the link from X to C. Node X is the guard node for link from P, Q, or itself to C. If node A needs to establish a new session key with node B, according to the protocol in Section 2.7, it sends the *Envelop* key to C. Node C receives the *Envelop* through one of its neighbors, say P. The guard nodes of the link from P to C (X and Y) overhear the *Envelop* that P forwards to C and buffers it to monitor what C does with it. If a guard node, say X, does not hear C forwarding the packet to the appropriate next hop within a certain time interval, it degrades its opinion of C

by incrementing *MalC*(*X*,*C*). The receiver collision [46], which occurs when the receiving node does not receive the packet due to collision, is alleviated using *MAC* layer acknowledgment. The ambiguous collision [46], which prevents a node *A* from hearing if a node *B* has forwarded a packet, due to a collision at *A*, is alleviated by employing multiple guards.

It is more involved to detect if *C* forwards a garbage packet instead of the *Envelop*. Since the communication from *A* to *C* and *C* to *B* are both encrypted, the guards cannot observe the traffic. To solve the problem, *A* appends the hash of the *Envelop* to the packet. The hash is compared by *C* and if correct, re-appended to the packet before forwarding to *B*. The guards can observe the hash values coming in and out of *C* and suspect *C* if the incoming and the outgoing hash values are different. If, however, the values are identical and the destination *B* detects a mismatch, then *C* is considered suspicious by *B*. This enables nodes other than the guards of the control node to share in monitoring the control node. When the *MalC*(*X*,*C*) reaches a pre-determined threshold value, $T_{counter\_threshold}$, it sends an alert to the base station. $T_{counter\_threshold}$, is calculated to account for natural failures, such as node and link failure errors. The calculation of the threshold $T_{counter\_threshold}$ needs further investigation and is not within the scope of this paper. A separate publication [47] explores this issue.

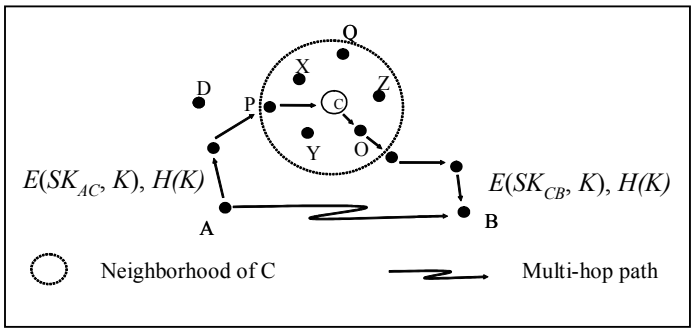

Figure 8: Example for Detection of Masquerading Nodes

We now analyze the protocol to update the malicious counter. To do that we use a scheme inspired by the idea of *degree of attack guilt* [40]. Each monitor may detect an event with a certain level of assurance, $L_{assurance}$, which lies between zero and one. A value of zero implies that the event is not considered suspect by the observer, while a value of one indicates that the observer is convinced that the event is a malicious event. The exact value is a function of the event and the observer (a guard node or a destination node). For example, if the control node modifies the *Envelop* but keeps the same hash value, then this does not appear as a malicious event to a guard node, but is a definite malicious event to the destination since the hash value does not match the packet content. Thus, the $L_{assurance}$ value at the guard node is zero, while that at the destination is one. For each event detected, the monitor α increments its $MalC(\alpha,C)$ by the result of multiplying $L_{assurance}$ by the maximum value of the counter $MalC_{max}$ (255 for an 8-bit counter). This implies that if the α is not certain ($L_{assurance} = 0$) about the ongoing monitored event, it does not increment $MalC(\alpha,C)$ and therefore does not degrade its opinion of node *C*. If however, node α is almost definite, then the increment will be close to $T_{counter\_threshold}$, thus taking the $MalC(\alpha,C)$ value above the threshold with a high likelihood. This in turn leads to detection of the malicious node.

When the $MalC(\alpha,C)$ crosses the threshold value $T_{counter\_threshold}$, α sends a message to the base station carrying the counter value and the malicious node's ID. However, in sensor networks where nodes may be compromised easily, it is clearly undesirable to base a decision on the input of only one other node. Therefore, the base station waits for a short time, $T_{suspect\_collection}$, to allow other nodes that should have noticed the same malicious event to send in their opinions. If the base station does not receive these alerts, it polls the corresponding nodes directly to send their *MalC* values. The base station considers the node to be malicious if a weighted majority of the polled nodes agree. This majority reduces the likelihood of blackmailing in which a compromised node falsely accuses a good node to degrade its trust level. The trust level, L_{trust}, of each node is a value between zero and one, where zero represents a mistrusted node and one represents a fully trusted node. The trust level is initialized to one. This is used as the weight in the calculations at the base station. The trust level for a node, say *B*, is calculated as

$$L_{trust}(B) = 1 - \left\{ \sum_{J} \frac{L_{trust}(J) \cdot MalC(J,B)}{N_m(B) \cdot MalC_{max}} \right\} \quad (1)$$

Where $L_{trust}(J)$ is the trust level of node *J* and $N_m(B)$ is the number of monitors of node *B* that report their *MalC* values to the base station. The sum is taken over each observer, *J*, of node *B* that reports its malicious counter value to the base station. This formula computes the weighted average of the malicious counter values. The weights in calculating the average are the trust levels of the nodes that report their malicious counter values.

The base station decides whether the node under investigation is malicious or not based on the trust level of the node. If the trust level goes under a pre-determined threshold value, T_{trust_level}, the base station declares the node as a compromised

node. Each neighbor of the malicious node is informed of the event. In response, each neighbor drops the malicious node from its neighbor list and ceases to forward its packets.

If a certain fraction of nodes erroneously report a control node to be suspicious, the base station may degrade their trust level through a mechanism such as shown by us in [47]. A table summarizing the timers and the threshold values used in SECOS and their effects on the protocol is presented in the Appendix.

3 Security Analysis

In this section, we discuss the ability of SECOS to deal with the three major classes of security attacks – confidentiality violation, denial of service attacks, and authentication violation.

3.1 Confidentiality Attacks

The key exchange protocol between two end points of a communication is described in Sections 2.7 and 2.8. We now show that this key exchange protocol does not reveal the shared key between two legitimate nodes irrespective of the number of compromised nodes if either of the following features is used. Note that these features are individually sufficient but not necessary for the proposition to hold.

1. The initial message $E(K, \widetilde{K})$ sent by the initiator of the key exchange, A, to the destination, B, cannot be obtained by the control node, *or*
2. The two parties involved in the key exchange, A and B, share an old session key in addition to $\widetilde{K}$ and use a combination of the new and previous session key for the communication. For example, if the previous session key was $\overline{\overline{K}}$, then A uses $\widetilde{K} \oplus \overline{\overline{K}}$ as the current session key for communication with B. In case a previous shared session key is not available, nodes A and B must establish the session through the secure base station and not through the control node.

Proposition: Under feature 1 ***or*** 2 above, it follows that compromising ***any*** number of nodes other than the two end-points does not reveal the shared key between them. This proposition holds even if the control node for the two end points is compromised.

Proof:

Case1: If feature number 1 is valid, then B is the only node in possession of the encrypted packet holding the key $E(K, \widetilde{K})$. Thus, the control node, C, does not have it and though it has K, it can never obtain the shared key $\widetilde{K}$.

Case2: If feature number 2 holds, the proposition can be proved using mathematical induction as follows.

Base case: Let the number of compromised nodes in the network be N_C. If $N_C = 0$, there is no compromised node and the claim is trivially satisfied. If $N_C = 1$, this compromised node could be either the control node of A and B or any other node. If it is not the control node then the session can not be disclosed since only the control node, other than A, can decrypt the packet holding the *Envelop*. Consider that the single compromised node is the control node. Two cases are possible. (1) Nodes A and B have a previous shared key using an old control node. The current compromised control node does not know this key because the old control node was not compromised since the current control node is the only compromised node in the network by assumption. (2) Nodes A and B do not have a previous shared old key so they use the secure base station to start up the shared key and not the compromised control node. In both cases 1 and 2, the compromised control node cannot disclose the secure session between A and B.

Inductive step: Assume that the session between A and B is secure under $(N_C - 1)$ compromised nodes, we want to show that it will be secure when a new node gets compromised for a total of N_C compromised nodes.

Inductive proof: If the N_C^{th} compromised node is not the control node, the claim is trivially satisfied. If the N_C^{th} node is the control node, then as in the base case, two cases are possible. (1) Nodes A and B share an old key (K_{old}), or (2) nodes A and B do not share an old key. In case (1), by the induction hypothesis, none of the $(N_C - 1)$ compromised nodes know the key, K_{old}. The new compromised node does not know K_{old} since the key was exchanged before the node got compromised. So if the new key exchanged through the compromised control node is K_{new}, then the new session key will be ($K_{old} \oplus K_{new}$). While the compromised node can know K_{new}, it cannot know K_{old}. In case (2), nodes A and B do not share an old key and hence obtain their key directly from the secure base station. This exchange is done using the shared session key with the base station and therefore the key is unknown to the control node. This completes the proof of the proposition.

Comments: The proof excludes the following cryptanalysis scenario. Assume the two nodes A and B have the startup key K_{old} from the main base station and then they use the K_{new_1} from control node C_1, K_{new_2} from control node C_2, …, K_{new_m} from control node C_m. An attacker may capture the packet holding K_{old} and crypt-analyze it to obtain K_{old}. By the time this is done, the control node is C_m. Then the session key at that time will be $K_{old} \oplus K_{new_1} \oplus K_{new_2} \oplus \ldots \oplus K_{new_m}$. To know this key, the attacker must either compromise all the control nodes C_1 up

through C_m or crypt-analyze all the packets holding the keys K_{new_1} up through K_{new_m} . It is expected that *m* will be a large number due to the small number of cipher packets the adversary has to crypt-analyze a key. It will be practically infeasible to compromise selectively all the control nodes C_1, …, C_m, especially considering that control nodes are pseudo-randomly chosen from among the ordinary sensor nodes. Alternatively, it will be practically infeasible to crypt-analyze all the keys K_{new_1}, …, K_{new_m}.

However, it is possible, though difficult, that neither of the features mentioned above is satisfied. In feature 1, the control node may be able to buffer all packets between *A* and *B*, either directly or with the help of a malicious colluding nodes, decrypts them and thus acquires $\tilde{K}$. Even if the communication of the initial message and the *Envelop* are randomized in time and order, it is possible that *C* buffers all messages within a window. Feature 2 is violated if the two parties do not share an old key and are unwilling to initiate key exchange using the main base station, possibly because it is far from either party. Section 3.1.1 presents a mathematical analysis of the probability of disclosing the secure session between *A* and *B* under certain number of *compromised nodes* if ***neither*** of the above features is used.

3.1.1 Probability of Secure Session Disclosure

In this subsection we provide a mathematical analysis of the probability of compromising the link between two arbitrary nodes *A* and *B* lying in the same control group with the number of compromised nodes in the network being a parameter. For the purpose of comparison with other key management protocols, we assume in this analysis that only compromised nodes may exist in the network (no external malicious nodes). We perform the analysis for SECOS, SPINS (a representative Kerberos like protocol), and a protocol by Du *et al.* [19] (a representative key pre-distribution protocol), and compare the results. We assume that SPINS has as many base stations as the number of control groups in SECOS (N_B) and that the nodes are uniformly distributed in the sensor field.

For the mathematical analysis, we use a restricted version of SECOS which does not use the two features mentioned in Section 3.1, i.e., the node does not use the multiple keys from previous control nodes or the communication with the base station and the control node may overhear communication between the two nodes in its control group. This serves as a plausible operating region for the protocol where resources are constrained, the control group size is small, or the control node colludes with a neighbor of the source-destination pair. The restriction on SECOS also serves to shed light on the advantages obtained by a specific feature of SECOS, namely using two packets – K($\tilde{K}$) and the Envelop for key exchange between two arbitrary nodes. Note that if we use the unrestricted version of SECOS, the analysis would become trivial since the probability of compromising the link between an uncompromised source-destination pair would be zero.

To disclose the session key between *A* and *B*, an attacker must obtain both the *Envelop* (*K*) and the packet that is sent directly from *A* to *B* ($E(K, \tilde{K})$). To obtain the *Envelop*, the control node for *A* and *B* must be compromised. To analyze the probability of capture of $E(K, \tilde{K})$, we create a *bounding path* between *A* and *B* which is the rectangular bounding box containing nodes that may overhear the communication from *A* to *B*. This is shown by the dotted box in Figure 9. This is an overestimate since we use a square that circumscribes the circular transmission range of a node. To capture $E(K, \tilde{K})$, there must be at least one node in the bounding path from *A* to *B* that is compromised (we assume no compromised nodes exist in the network). Let the average number of hops between a pair of nodes in the control group be H_{ctrl}, the density of nodes in the sensor field be *D*, and the communication range be *R*. The probability of capturing $E(K, \tilde{K})$is less than or equal to the probability of having at least one compromised node in the bounding path. Let *N* be the total number of nodes in the sensor field and $SG_{ctrl}=N/N_B$ is the size of a control group. Let the number of compromised nodes in the network be N_C and assume that the compromised nodes are uniformly distributed in the field. Let E_2 represent the event that there is at least one compromised node in the bounding path.

The identity of the current control node in a control group can be easily deduced by an attacker. However, as mentioned in the assumptions, it takes a finite amount of time T_{comp} to compromise a node. The period of rotation of the control node is smaller than T_{comp}. Thus, starting from an uncompromised network, it will be impossible for an attacker to compromise the control node after identifying it. So the attack model for the analysis is that the attacker randomly picks a node to compromise. Let E_1 be the event that this randomly chosen node is a control node, for some arbitrary source-destination pair A and B.

$$P(E_1) = \frac{\#Compromised\ Nodes}{\#Nodes\ in\ Network} = \frac{N_C}{N} \tag{2}$$

The probability of compromising the link between *A* and *B* ($P_{C(A-B)}$) is

$$P_{C(A-B)} = P(E_1E_2) = P(E_2 \mid E_1)P(E_1) \tag{3}$$

The number of nodes within the bounding path N_{bp} is given by its area times the density of nodes in the network.

$$N_{bp} = (H_{ctrl}+1)R \cdot 2R \cdot D = 2(H_{ctrl}+1)R^2D \tag{4}$$

Let E_3 be the event that the control node lies in the bounding path. Then the probability of E_3 is

$$P(E_3) = \frac{N_{bp}}{SG_{ctrl}} \tag{5}$$

Note that in the previous formula we consider the size of the control group since A and B lie within the same control group.

$$P(E_2 \mid E_1) = P(E_2 \mid E_1 E_3)P(E_3) + P(E_2 \mid E_1 \bar{E}_3)P(\bar{E}_3) \tag{6}$$

Let $N_G = N\text{-}N_C$ represents the number of uncompromised (good) nodes in the network. The number of ways in which we can choose N_{bp} good nodes is

$$\binom{N_G - 2}{N_{bp} - 2} \tag{7}$$

The total number of ways in which we can choose N_{bp} nodes is

$$\binom{N - 2}{N_{bp} - 2} \tag{8}$$

Since A and B both are assumed to be non-compromised nodes, they are subtracted from N_{bp}, N_G, and N.

$$P(E_2 \mid E_1 \bar{E}_3) = 1 - \left[\binom{N_G - 2}{N_{bp} - 2} \Big/ \binom{N - 2}{N_{bp} - 2} \right] \tag{9}$$

$$P_{C(A-B)} = \left(\frac{N_C}{N}\right)\left\{\left(1 - \left[\binom{N_G - 2}{N_{bp} - 2} \Big/ \binom{N - 2}{N_{bp} - 2}\right]\right)(1 - P(E_3)) + 1.P(E_3)\right\} = \left(\frac{N_C}{N}\right)\left\{\left(1 - \left[\binom{N - N_C - 2}{2R^2 D(H_{ctrl} + 1) - 2} \Big/ \binom{N - 2}{2R^2 D(H_{ctrl} + 1) - 2}\right]\right)(1 - P(E_3)) + P(E_3)\right\} \tag{10}$$

In SPINS [1], which represents an example of the Kerberos-like protocols, the base stations are fixed. In order to make the sensor network economical, the authors assume that the base stations are not equipped with any specialized mechanisms or hardware to prevent compromise. They only assume that the base station has sufficient battery power to surpass the lifetime of all sensor nodes, sufficient memory to store cryptographic keys, and means for communicating with outside networks. Therefore the base stations in SPINS are equally likely to be compromised as any other sensor nodes. The model for the adversary is that it *can* target the base stations for compromising them. The attacker can identify the base stations and they are fixed so the adversary has enough time to try to compromise them. Thus

$$P_{C(A-B)} = \begin{cases} \frac{N_C}{N_B} & \text{if } N_C < N_B \\ 1 & \text{if } N_C \geq N_B \end{cases} \tag{11}$$

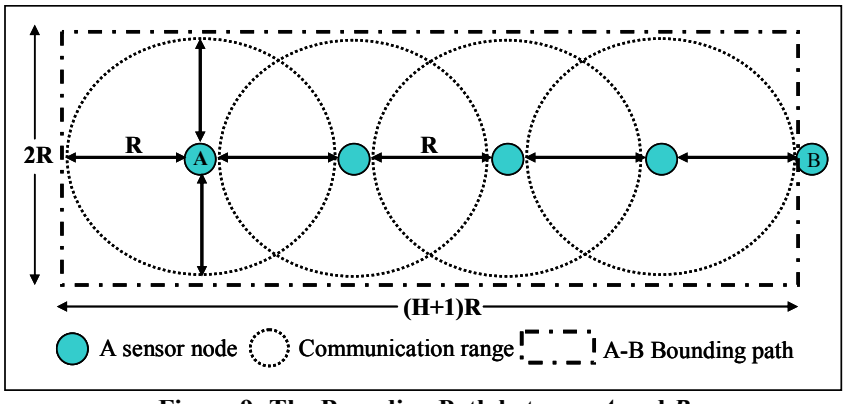

Figure 9: The Bounding Path between *A* and *B*

The protocol by Du *et al*. [19] represents an example of a key-pre-distribution scheme and is summarized by us in Section 6. The authors present a corresponding calculation of $P_{C(A-B)}$ as

$$P_{C(A-B)} = \sum_{i=\delta+1}^{N_C} \binom{N_C}{i} \left(\frac{\tau}{\omega}\right) \left(1-\frac{\tau}{\omega}\right)^{N_C-i} \tag{12}$$

Where δ is the key space threshold, i.e. compromising (δ+1) nodes will compromise the whole key space. ω is the size of the key space's pool, i.e. there are ω key spaces for each node to pick from. τ is the number of different key spaces that each node holds. The memory requirement at each node is $mem = (\delta+1)\times\tau$. Also, they provide the formula for the probability that any two neighboring nodes can establish a secure session between them as

$$P_{actual} = 1 - \frac{\left((\omega-\tau)!\right)^2}{(\omega-2\tau)!\,\omega!} \tag{13}$$

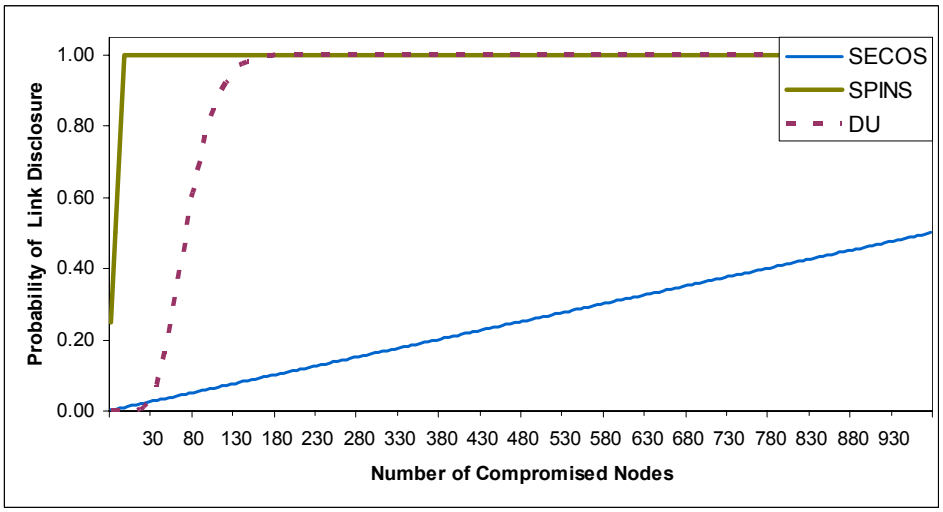

Figure 10: Probability of compromising a randomly selected link between two uncompromised nodes as a function of the number of compromised nodes in the network.

Figure 10 shows the comparison among these three schemes (SECOS, SPINS, Du) using: ω = 50, *mem* = 200, τ=5, and *Pactual*=0.42 as parameters for Du's scheme (δ is calculated as 39 based on the memory constraint *mem*), N_B = 20 for SPINS, and. N = 2000, R = 30, D = 15 neighbors for each node, and H_{ctrl} = 10 as parameters for SECOS. Notice that Du's scheme has only 0.42 for P_{actual} while SECOS and SPINS both have 100% probability for any two nodes to establish a secure session between them. According to Figure 10, SECOS has lower probability of compromising a link than the other two protocols over a large range of the operating region. The probability goes to one for SPINS when the number of compromised nodes is greater than the number of base stations. Also, the link disclosure probability goes to one for Du's scheme when the number of compromised nodes is greater than the δ threshold. However, for a small number of compromised nodes, Du's scheme is the most robust.

3.2 Denial of Service (DoS) attack

1. *DOS attack against a control node.* This may be launched through a compromised node when it repeatedly asks the control node for forwarding a key. This kind of attack is handled by keeping a state vector at the control node for the currently active nodes that have recently requested key forwarding, and ignoring and sending feedback to the base station if a node behaves abnormally, e.g., asking for keys to communicate more than the feasible data rate. The feasible data rate is determined using a running window of the last *m* key requests and considers the communication bandwidth and the key cache size.
2. *DOS attacks by a compromised control node:* We reduce the probability of the presence of a compromised control node by a judicious selection of the control node based on trust level by periodically changing the control node. However, for the time period when a compromised node serves as a control node, it can prevent two legitimate nodes, *A* and *B*, from establishing a common key between them. In such a situation when the initiator cannot establish the secure session using the control node, it can perform the key exchange using the base station as an intermediary. Each of *A* and *B* share a session key with the base station, which is distinct from the shared session key with the control node, and this can be used to establish a secure channel. This solution is also valid when control node is unavailable due to a natural failure. The base station verifies that the requests for *Envelop* forwarding are coming from a legitimate node in the network and if it finds the control node is non-existent, installs a new control node. This scheme is identical to that used in SPINS in the general no-attack case.

Control node monitoring results in detecting the control node if it launches a DoS attack. To see this, consider the following two possible DoS attacks that a malicious control node could launch. In the first DoS attack, the control node refuses to forward the *Envelop* it received from the source to the intended destination. This is an easy attack to detect and can be detected by both the guards and the destination. The guards see a packet entering the control node but no corresponding packet sent out. The destination detects the attack since it does not get the *Envelop* though it receives the communication from the source. However, the assurance value of the guards is higher than that of the destination. At the destination there is a possibility that the *Envelop* is lost in the path from the source to the control node or from the control node to the destination. So the opinion counter at a guard is incremented by a value greater than that at the destination. The determination by the guard is still not full-proof since it is possible that the error is in the last hop to the control node or first hop out of the control node. Let P_{Lerr} represent the probability of natural error in a packet on one link. Let the number of hops the *Envelop* traverses from the control node to the destination be H_{com}, and the average number of hops in the same control group be H_{ctrl} (the number of hops between the source and the control node), then the probability of incorrect reception at the destination is

$$Drop_{natural} = 1 - \left(1 - P_{Lerr}\right)^{\left(N_{com} + N_{ctrl} - 2\right)} \tag{14}$$

Let P_{CD} represents the probability that a node is compromised and dropping packets. The probability that the packet will not reach the destination due to a malicious node other than the control node is

$$Drop_{malicious} = 1 - \left(1 - P_{CD}\right)^{\left(N_{com} + N_{ctrl} - 2\right)} \tag{15}$$

Then assurance value of this malicious event at a guard is $L_{assurance}$ = 1-2P_{Lerr}, and at the destination is

$$L_{assurance} = 1 - \left(Drop_{natural} + Drop_{malicious}\right) = \left(1 - P_{Lerr}\right)^{\left(N_{com} + N_{ctrl} - 2\right)} + \left(1 - P_{CD}\right)^{\left(N_{com} + N_{ctrl} - 2\right)} - 1 \tag{16}$$

In the second DOS attack, the control node forwards a modified *Envelop*, either by modifying the *Envelop* while keeping the same hash value associated with it, or by modifying both the *Envelop* and the hash value. The technique to circumvent the two types of DoS attacks are discussed in detail in Section 2.9.

3. *DoS attacks against regular nodes:* It is relevant to talk of only those DoS attacks against regular nodes that are enabled by mechanisms in SECOS. One possible DoS attack that may be launched against a legitimate node, *B*, is storage exhaustion by sending garbage packets to *B*, which buffers it in the expectation that the key needed to decrypt the packet is forthcoming from the control node. Requiring *B* to limit the number of unencrypted packets received from a specific source, accompanied by the inability of that source to launch an ID spoofing attack due to the neighbor watch (Section 3.3) alleviates this attack.

3.3 Authentication Attack

Another possible class of attacks is The ID spoofing and Sybil attacks in which a node impersonates other nodes [54] [55]. Through this attack, a compromised node can obtain knowledge of shared keys between other nodes. This class of attacks may be launched by a compromised control node, a regular node, or multiple nodes in collusion. SECOS handles the problem of regular nodes trying to masquerade as the control node by providing the control node challenge mechanism (Section 2.5) and for control nodes trying to masquerade as a different sensing node by using local monitoring (Section 2.9). The two kinds of authentication attack whereby a node impersonates a neighboring node or a non-neighboring node are detected by the neighbor watch mechanism by the neighbors of the compromised node according to the scheme described in Section 2.9. Note also that many key management protocols (e.g. [1],[11]) do not address the authentication problem. Key management protocols in [17] and [19] are examples which address authentication as an inherent property of their protocol.

If the control node, *C*, is compromised, it may launch the following attack to uncover the key between two nodes in its control group, *A* and *B*. Node *C* sends to *B* a key $\widetilde{K}$ encrypted using the *Envelop K* claiming that it is from *A*. Node *C* performs the same communication with *A*, claiming it is from *B*. Then *C* sends the *Envelop K* to both *A* and *B* after encrypting it with the respective session keys. The communication between *A* and *B* is now under the control of *C*. In SECOS, this attack is prevented through two mechanisms – local monitoring. First, if *C* tries to impersonate *B* and sends a packet, any of its neighbors, which does not have *B* in its neighbor list detects this while *A* itself will not be able to detect the impersonation. So $L_{assurance}$ value for the guards will be one and it will be zero for the destination. Second, if *C* generates the spurious messages and claims it is forwarding the message from *B* through a neighbor, *O* in Figure 8 , this is detected by the nodes *Y* and *Z*, which are acting as the guard nodes for the communication through *O*, while it can not be detected by the destination, *A*. So $L_{assurance}$ value for the guards is one and it is zero for the destination.

We quantify the overhead in terms of control messages for each of the operations in SECOS, such as key establishment within and across control groups, neighbor watch, and control node monitoring. The analysis is presented in the Appendix.

4 Determining Control Group Size

In this section, we perform mathematical analysis to determine the optimal control group size in SECOS based on the constraints of the sensor network and the desired level of security. We introduce some notations for this analysis. The regular cache size at each node is S_C, the hit rate in the cache α_C, and the miss rate β_C =1-α_C. The control cache size is S_{CC}, and its hit and miss rates are α_{CC} and β_{CC}, respectively**.** The hit rate is the probability that an item is found in the cache while the miss rate is the probability that an item is missed from the cache. The control group size that is to be optimized is SG_{ctrl}, and the communication group size is SG_{com}. We introduce the communication group for a node as the neighborhood of that node, with which it *predominantly* communicates. The quantitative meaning of predominant is made clear in the particular discussion. For the analysis in this section, we assume that the communication happens completely within the communication group. Each node generates packets according to a Poisson process with rate $1/\lambda$. The destination is chosen at random from the communication group. The destination is changed once every μ seconds on an average, again using an exponential distribution. The control node has an average lifetime of T_{ctrl}. *S(Pkt*) gives the size of the *Pkt* packet. H_{com}, H_{ctrl}, and H_{all} are the average number of hops between nodes within the same communication group, between a node and the control node, and between a node and the base station. E_{nergy} gives the energy for transmission and reception of one bit. The summary and notations for some of the control packets used in SECOS are given in Table 1.

Packet Notation	Description	Packet Notation	Description
K_req	The *Envelop* from the source to the control node or from the control node to the destination.	*K_repf*	Relay the *Envelop* from one control node to another, used in inter-group key establishment
Data	Data packet	*K_rep*	The encrypted key from the source to the destination

Table 1: Summary of relevant SECOS packet types

4.1 Maximum Control Group Size

The maximum allowable size of the control group is determined by three factors–computational capabilities of the control node, bandwidth available around the control node, and the storage capacity for keys in the control node. These factors are discussed below. Here, G_{COMP} is the maximum control group size under the computational limitation only, G_{BW} is the maximum control group size under the bandwidth limitation only, and G_{STORE} is the maximum control group size under the storage limitation only.

1. *Computational Capabilities (*G_{COMP}). The computational capability of the control node to service key requests from nodes in its group is one of the factors that bound the control group size. Assume that the computational capability of the control node allows it to process *IP* instructions per second and the encryption algorithm for the *Envelop* encryption and decryption, the hash function computation, and the *MAC* encryption and decryption according to the steps shown in Figure 7(a) require *IK* instructions. The maximum number of keys that can be serviced *is IP*/*IK* keys per second. So if the node changes a destination every μ seconds and the miss rate in the regular cache is β_C, a request is generated by a single node once every μ/β_C seconds.

$$G_{COPM} \le \frac{IP \cdot \mu}{IK \cdot \beta_C} \tag{17}$$

2. *Channel Bandwidth (*G_{BW}*).* On average the available bandwidth for each node given channel bandwidth *BW* is BW/N_{nbr} where N_{nbr} is the number of one-hop neighbors of the node. Given the range of wireless transmission *(r)* and the density of nodes *(ρ): D = Л r*^{2} *ρ*. Part of this traffic bandwidth is consumed by data. Thus the available *BW* for control communication (BW_c) is the total bandwidth per node minus the amount of data traffic

$$BW_c = \frac{BW}{N_{nbr}} - \frac{2 \cdot S(Data)}{\lambda} \tag{18}$$

Each new session key served generates 2*S(K_req)* amount of traffic. Taking into account the regular cache misses and the key request rate this term is multiplied by (β_C.1/μ).

$$BW_c \ge G_{BW}(2 \cdot S(K\_req))(\beta_C / \mu) \Rightarrow G_{BW} \le BW_c / (2 \cdot S(K\_req)(\beta_C / \mu)) \tag{19}$$

3. *Storage Capacity (*G_{STORE}*).*The storage refers to the ring cache in the control node which stores the keys of nodes in the control group. If the storage requirement of each key is S_{Key} and the available flash memory for the ring cache is *FM*, then the storage upper bound is given by

$$G_{STORE} \le FM / S_{Key} \tag{20}$$

The maximum size of the control group is the minimum of those calculated from equations (21),(22), and (23) above.

$$G_{max} = \min(G_{COMP}, G_{BW}, G_{STORE}) \quad (21)$$

The previous three factors came from resource constraints. A fourth factor arises from the security requirement. This is the security tolerance (G_{SEC}) when a control node gets compromised. G_{SEC} represents the maximum size of the control group under a certain acceptable number of compromised sessions or exposed messages. It is assumed that all the sessions that are established after the control node is compromised are disclosed.

4. *Security tolerance* (G_{SEC}). We want to limit the amount of communication that will become exposed due to the control node being compromised. Let *N(s)* be the acceptable number of message communications that can be exposed. Let the rate at which nodes are compromised be λ_{SEC}. Consider a round as the time a control node maintains its privileged position. The length of a round is T_{ctrl}. Consider an infinitesimally small time slice *dt*, after time *t* has elapsed in a round. The number of nodes that can be compromised in this time slice is λ_{SEC}dt. In the worst case, all the compromised nodes are control nodes. As a result of compromising these control nodes, the number of communication sessions that will become exposed are $G_{SEC}.((T_{ctrl}-t)/\mu)\times\beta_C$. Integrating over the entire round, we have

$$\int_0^{T_{ctrl}} \frac{\lambda_{SEC} G_{SEC} \beta_C (T_{ctrl} - t)}{\mu} dt = \frac{\lambda_{SEC} G_{SEC} \beta_C}{2\mu} T_{ctrl}^{\ 2} \le N(S) \quad (22)$$

$$G_{SEC} \le \frac{2\mu N(S)}{\lambda_{SEC} \beta_C T_{ctrl}^{\ 2}} \quad (23)$$

The maximum size of the control group becomes,

$$G_{max} = \min(G_{COMP}, G_{BW}, G_{STORE}, G_{SEC}) \quad (24)$$

4.2 Energy-wise Optimal Control Group Size

Here we wish to find the optimal control group size based on security and energy concerns. For this analysis, we consider the energy consumed in the entire network per unit time, which is equivalent to the power requirement of the network. We want to increase the security by minimizing the time between control node refreshments and we want to decrease the overhead energy of the protocol. The security requirement favors decreasing the time to refresh the control nodes and the smallest is the best while a larger period is more optimal energy wise. So we will proceed to optimize the energy overhead. In doing so, we face two conflicting factors. The first is the number of nodes that can be served by the same control node, and the second is the average number of hops to the control node. The first factor favors increasing the control group size, since that will reduce the occurrence of the energy expensive inter-control group key setup communication. The second factor favors decreasing the control group size, since that will reduce the number of hops between a sensing node and the control node.

Three factors are to be considered for the overhead energy consumption of SECOS: the destination of the packet to be sent (whether within the same control group or outside), the probability of regular cache hit, and the probability of control cache hit. In the following derivation, we assume that the average number of hops between nodes is proportional to the number of nodes under the same density and traffic conditions, such that: $H_{ctrl} = \max(H_{com} \times SG_{ctrl}/SG_{com}, 1)$. From these we derive the following four cases:

Case 1: Hit in the regular cache. This occurs with probability α_C that can be calculated as follows:

$$\alpha_C = \frac{S_C \times \lambda}{SG_{com} \times \mu} + \left(1 - \frac{\lambda}{\mu}\right) \sum_{k=0}^{S_C} \left\{ \binom{N-1}{K} \left(\frac{1}{N-1}\right)^k \left(1 - \frac{1}{N-1}\right)^{N-1-k} \right\} \quad (25)$$

The term $(S_C \times \lambda)/(SG_{com} \times \mu)$ represents the probability that the key is found in the regular cache during the send of the first packet and the subsequent terms represent the probability that the second, the third, the fourth, etc packets hit. We assume that the size of the regular cache is greater than the number of packets sent in μ seconds. However, $\alpha_C = 1$ if the cache size is greater than the communication group size ($S_C > SG_{com}$). If there is a hit in the regular cache, no overhead energy is spent.

Weighted energy overhead = Energy overhead per miss. Probability = 0.

Case 2: Miss in the regular cache and the destination is in the same control group. The probability of regular cache miss is $\beta_C = 1 - \alpha_C$. The probability of communication within one control group is SG_{ctrl}/SG_{com}. If $SGctrl > SG_{com}$, i.e., the control group is larger than the communication group, then the communication is always within one control group and the probability is one.

Weighted energy overhead = Energy overhead per miss. Probability =

$$\left(2 \times S(k\_req) + S(k\_rep)\right) \times H_{ctrl} \times E_{nergy} \times \beta_c \times \left(\frac{SG_{ctrl}}{SG_{com}}\right) \quad (26)$$

Case 3: Miss in the regular cache, the destination is outside the control group and hit in the control cache. The probability of control cache hit, given that the number of control groups within the communication group is $N_{BC}=SG_{com}/SG_{ctrl}$, is given by: $\alpha_{CC} = S_{CC}/(N(SG_{com})-1) = S_{CC}/((SG_{com}/SG_{ctrl})-1) = SG_{ctrl}\times S_{CC}/(SG_{com}-SG_{ctrl})$. However, if $SG_{ctrl} > SG_{com}/(S_{CC}+1)$, $\alpha_{CC} = 1$.
Weighted energy overhead =Energy overhead per miss. Probability =

$$\left\{\left(2\times S(K\_req)+S(K\_rep)\right)H_{ctrl}+S(K\_repf)\times H_{com}\right\}E_{nergy}\times\beta_C\left(1-\frac{SG_{ctrl}}{SG_{com}}\right)\left(\frac{SG_{ctrl}\times SG_{com}}{SG_{com}-SG_{ctrl}}\right) \quad (27)$$

Case 4: Miss in the regular cache, the destination is outside the control group, and miss in the control cache .The probability of control cache miss $\beta_{CC} = 1 - \alpha_{CC} = 1 - SG_{ctrl}\times S_{CC}/(SG_{com}-SG_{ctrl}) = (SG_{com}-SG_{ctrl}-SG_{ctrl}\times SG_{com})/(SG_{com}-SG_{ctrl})$
Weighted energy overhead = Energy overhead per miss. Probability

$$\left\{\left(2\times S(K\_req)+S(K\_rep)\right)\times H_{ctrl}+S(k\_repf)\times H_{com}+2\times S(K\_req)H_{all}\right\}E_{nergy}\times\beta_C\left(1-\frac{SG_{ctrl}\times S_{CC}}{SG_{com}-SG_{ctrl}}\right)\left(1-\frac{SG_{ctrl}}{SG_{com}}\right) \quad (28)$$

The total overhead energy of the protocol equals the sum of the contributions of the above four cases. Let the size of the key reply be S_R, i.e. $S(K\_rep)= S_R$. And since the size of key request equals the size of key reply forward which is approximately three times the size of the key reply, we have $S(K\_req) = S(K\_repf) = 3S_R$. The total overhead energy T_E is written as several separate equations each for a region bounded by discontinuities:
If $SG_{ctrl} > SG_{com}$ then

$$T_E = 7\times S_R\times H_{ctrl}\times E_{nergy}\times\beta_C \quad (29)$$

If $SG_{ctrl} < SG_{com}$ and $SG_{com} < SG_{ctrl}\,(S_{CC}+1)$ then

$$T_E = \left\{7\times S_R\times H_{ctrl}\times E_{nergy}\times\beta_C\frac{SG_{ctrl}}{SG_{com}}\right\}+\left\{(7\times S_R\times H_{ctrl}+3\times S_R H_{com})\times E_{nergy}\times\beta_C\left(1-\frac{SG_{ctrl}}{SG_{com}}\right)\right\} \quad (30)$$

If $SG_{com} > SG_{ctrl}\,(S_{CC}+1)$ then

$$\begin{aligned}T_E = {} & \left\{7\times S_R\times H_{ctrl}\times E_{nergy}\times\beta_C\frac{SG_{ctrl}}{SG_{com}}\right\}+\left\{(7\times S_R\times H_{ctrl}+3\times S_R\times H_{com})\times E_{nergy}\times\beta_C\left(1-\frac{SG_{ctrl}}{SG_{com}}\right)\left(\frac{SG_{ctrl}\times S_{CC}}{SG_{com}-SG_{ctrl}}\right)\right\}\\ & +\left(7\times S_R\times H_{ctrl}+3\times S_R\times H_{com}+6\times S_R\times H_{all}\right)E_{nergy}\times\beta_C\left(1-\frac{SG_{ctrl}}{SG_{com}}\right)\left(\frac{SG_{ctrl}\times S_{CC}}{SG_{com}-SG_{ctrl}}\right)\end{aligned} \quad (31)$$

We substitute $H_{ctrl} = 1$ when $SG_{ctrl}\times H_{com} < SG_{com}$ and $H_{ctrl} = SG_{ctrl}\times H_{com} / SG_{com}$ when $SG_{ctrl}\times H_{com} \geq SG_{com}$ in the above set of equations.

By minimizing T_E with respect to SG_{ctrl}, we get a value of $SG_{ctrl} = G_{energy\_opt}$ that minimizes the overhead energy of SECOS. This does not give a closed form solution since there are discontinuities due to α_C, α_{CC}, and H_{ctrl}. The equation can be solved numerically as shown below.

If the above analysis gives a control group size that is smaller than the maximum size calculated in Section 4.1, then we choose that. Else, we are bounded by the maximum control group size. Mathematically, the chosen control group size is $SG_{ctrl} = \min(G_{energy\_opt}, G_{max})$.

Figure 11 presents a numerical solution for the optimal control group size for optimizing the total power consumption for a network of 2000 nodes with $H_{all} = 100$, $H_{com} = 10$, $SG_{com} = 200$, $\beta_c = 0.2$, $E_{nergy} = 100$ pJ, $S_R = 128$ bit, and three different values for S_{CC} 1, 4, and 9. As Figure 11 shows, the optimal group size occurs when $SG_{ctrl} = SG_{com}/(S_{CC}+1)$. The consumed power starts very high for small control group sizes relative to the communication group size because a large portion of the communication goes through the costly inter-group communication. As the control group size increases, the power decreases due to the decrease in the inter-group communication to the point where the number of control groups within the communication group equals the size of the control cache. Thus, decreasing the number of control groups, by increasing the control group size beyond this point does not provide any additional gains since all inter-group communication hits in the control cache. Increasing the control group size after this point starts increasing the power linearly due to the increase in the average number of hops to the control node within the same control group. In our analysis, the increase in the number of hops is assumed to be linear with the size of the control group.

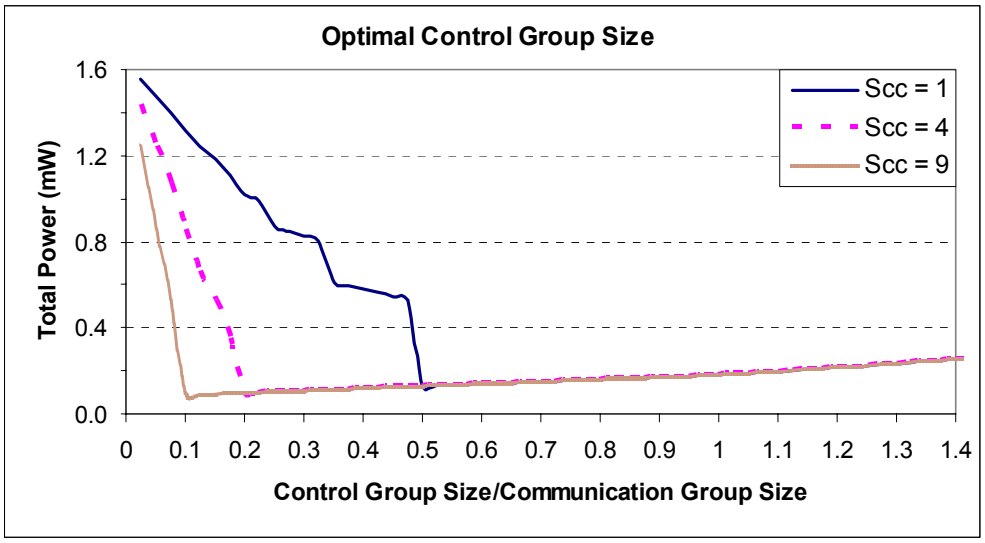

Figure 11: Total power consumed in SECOS with varying control group size.

5 Experiments & Results

We build simulation models for SECOS and SPINS using the network simulator, ns-2. We generate a grid topology for the sensor field and distribute the nodes randomly on it. We distribute the nodes into control groups based on geographical location and place the base station at the top right corner of the field. We simulate 9 different communication patterns by changing the communication group size and the average percentage of communications that go within that group, for example 90/10 communication means that 90% of the destinations are chosen from within the communication group while the rest are picked randomly from the whole network. Four different values of the relative size of the communication and control group are chosen for the experiment – 0.5, 1, 2, and 4. The simulation parameters used are shown in Table 2.

Bandwidth	40 Kbps	Control group size (SG_{ctrl})	10
Transmission range in meters	50	Ring cache size	20
Number of nodes in the sensor field	200	Regular cache size (S_C)	0,5,10
The topology in square meters	120X600	Simulation Time	10^5 s
Frequency of destination change (μ)	20 s	Frequency of control node change (T_{ctrl})	200 s
Frequency of packet generation (λ)	5 s	Frequency of session key refreshment	200 s
Number of control groups	20	Control cache size	5

Table 2: Simulation Parameters for Evaluation

We measure two parameters for both SECOS and SPINS: the total overhead energy due to key management and the average end-to-end delay of data packets. The end-to-end delay of a data packet is the sum of the delay of key management and data transmission delay. For the plots, we use the ratio of the SPINS value to the SECOS value. A higher value on the plot implies better performance by SECOS with a value of one being the crossover point.

In the first experiment, we vary the size of the regular cache at each sensing node and observe the output parameters for 4 different sizes of the communication group. The 100%:0% and 90%:10% communication patterns show identical trends but the 90%:10% case is less favorable to SECOS because occasionally the destinations could be far, outside the control group. Focusing on the less favorable 90%:10% case, we show the results in Figure 12(a) and (b).

Note that in these results, the two energy consuming but security enhancing parts of SECOS are simulated, namely, the periodic refreshment of the session keys, and the periodic change of the control node. From these graphs we find that SECOS outperforms SPINS both in terms of saving energy and reducing end-to-end delay. SECOS reduces the energy consumption by a factor ranging from 1.2 to 5.7, depending on the communication pattern and the cache size.

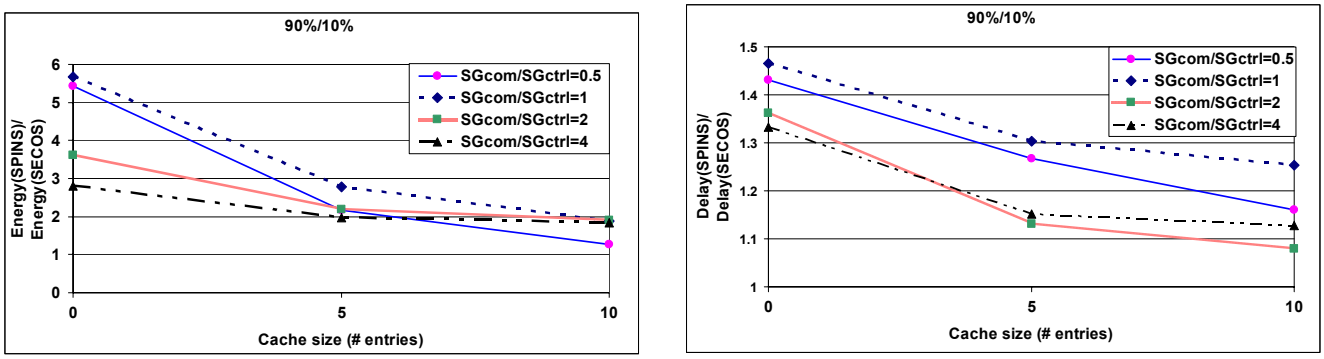

Figure 12: Ratio of (a) overhead energy expended and (b) end-to-end data latency for SPINS and SECOS with varying cache sizes for different communication group sizes

If the cache can store the keys of all the nodes that a node may communicate with, SPINS performs comparably in energy to SECOS. But this is inadvisable from the point of view of forward security since a number of old sessions may be exposed if the node gets compromised. If we use the most secure configuration with no cache, SECOS has a 2.8-5.7 fold energy reduction. As the cache size increases, the need for key exchange decreases and thus the difference between SECOS and SPINS decreases until the point when the cache can hold all the needed keys. For the simulation parameters here, the maximum benefit to SECOS is when the control group size equals the communication group size. As the communication group size increases beyond this, SECOS is favored less and less. The difference between SECOS and SPINS decreases as more inter-group communication takes place and this process is more energy consuming in SECOS than in SPINS. However, a reasonable sized control cache as used in these experiments still ensures that SECOS performs better than SPINS. This is explained by the fact that the control cache eliminates the necessity of a control node to create a new secure channel with another control node using the base station as the intermediary for every inter-group communication. It is seen that the difference between SECOS and SPINS decreases more sharply for SG_{com}/SG_{ctrl}=0.5 and 1. This is due to the fact that for these ratios, SECOS initially far outperformed SPINS with small cache sizes. The trend in delay is identical to that for the energy overhead. The reason behind the lower energy consumption is that the number of hops to exchange the keys is lower, which translates directly to a lower delay.

Next, we consider the communication pattern where any node can talk to any other node in the sensor field, which is referred to as all-to-all communication. The results are shown in Figure 13(a). In all-to-all communication, the energy ratio decreases as the cache size increases for a reason similar to that in the other communication patterns. However, it is seen that the reduction becomes flat beyond 10 cache entries. With 20-entry control cache, which effectively mimics an infinite cache, SECOS consumes 58% less energy and incurs 8.8% less delay. This indicates that even if the possibility of a sensing node being compromised can be disregarded, and the cache size made arbitrarily large, SECOS outperforms SPINS. This is explained by the fact that relative to the number of control groups in the entire network, the control cache is large enough that SECOS does not have to resort frequently to the expensive inter-group communication. In a real-world deployment, it is likely that the communication group of a node will not span too many control groups, since a node is unlikely to communicate frequently with nodes geographically very distant from it. Therefore, with reasonable control cache sizes, SECOS will perform well.

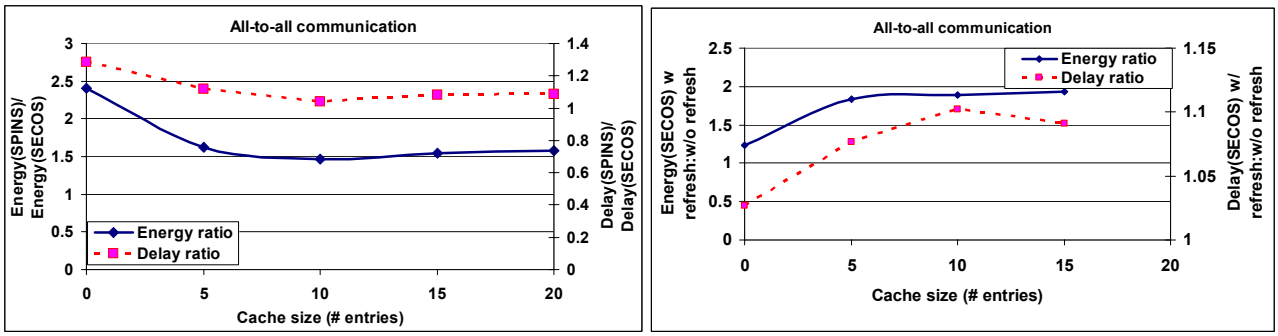

Figure 13: Ratio of overhead energy and delay for (a) SPINS: SECOS (b) SECOS with key refreshment and control node change: SECOS without these techniques

Finally, we bring out the overhead SECOS incurs due to two mechanisms for improving security, namely refreshment of session keys, and change of the control node. Figure 13(b) shows that the energy overhead of SECOS is 25% compared to SECOS-no-refresh when there is no cache. Relative overhead of SECOS with respect to SECOS-no-refresh increases as the cache size increases since SECOS increasingly sees the performance impact of purging the cache. At higher cache sizes,

93% energy may be saved if refreshment and control node change are suppressed. The reduction in delay is about 9% at high cache sizes.

6 Related Research

SECOS uses the well-known technique of node clustering. Node clustering is a technique that has been used in different areas in sensor networks. Secure data aggregation [25], self-assembling deployment and configuration of large number of nodes [41], energy saving for data aggregation [42], power optimal routing [43], control and management of routing protocols [44], and energy and communication cost optimization [45] present examples of these areas.

It is well accepted that asymmetric key cryptography is not well suited to sensor networks because of high computational expense. Hence, asymmetric key algorithms for key management in sensor networks ([3],[4],[5] for survey) look infeasible except under energy rich environments. Symmetric key techniques appear better suited for sensor networks. Different flavors of symmetric key techniques have been used. Some of these flavors either rely on a common shared secret key between all the nodes leading to a relatively insecure deployment, or have a separate shared key between each pair leading to a large amount of key storage for the large-scale sensor networks we are targeting. Examples of these protocols are the pre-deployed keying with variations of group-wise pre-deployed keying, secret sharing pre-deployed keying, and k-Secure t-limited group-wise pre-deployed keying [6],[7],[11],[13]. The requirement of keeping radio communication minimal makes many of the proposed purely symmetric algorithms impractical since they add a fixed size overhead number of bytes to a small payload packet [8],[10].

A large number of key management protocols for sensor networks fall in the category of key pre-distribution [2],[11],[15],[17],[18],[19],[21],[22],[23],[24],[28],[29]. Eschenauer and Gligor [11] present a key management scheme for sensor networks based on probabilistic key pre-deployment. They use a large pool of keys from which they select *m* keys at random, which are loaded into each sensor node before deployment. In order to communicate, any two nodes either use a common key they share. If such a common key does not exist, a series of intermediate nodes, which pair-wise have a common key, are used to exchange a key securely. However, compromising any node reveals all the keys in the node. This may compromise communication between other nodes that may use a shared key, which happened to be within the keys of the compromised node. Furthermore, the key establishment process is open to compromise since the identifiers are broadcast to a receiver set that has not yet been authenticated. Chan *et al.* [2] extend this scheme by requiring more than one key to be shared between any two nodes to establish a secure communication. They also use partial key exchanges on multiple paths to ensure security from some nodes on the path being compromised. Its major drawback is that it adds substantial overhead in finding multiple disjoint paths and a larger fraction of nodes than [2] may not be able to establish secure sessions with each other. Zhu *et al*. [28] present an approach for establishing a pair-wise key that is exclusively known to a pair of nodes with overwhelming probability, based on the combination of probabilistic key sharing and threshold secret sharing.

In [15], Blom proposes a key pre-distribution scheme that allows any pair of nodes to find a secret pair-wise key between them. Compared to the (N-1) pair-wise key pre-distribution scheme, Blom's scheme only uses δ+1 memory spaces with δ much smaller than N. The tradeoff is that, unlike the (N-1) pair-wise key scheme, Blom's scheme is not perfectly resilient against node capture. On one hand if (δ+1) nodes are compromised all pair-wise keys of the entire network are compromised. On the other hand, as δ increases, the computational and storage overhead increase, which make the scheme unscalable. Du *et al.* [19] extend the work done by Blom in a manner motivated by the proposed q-composite extension [2] of the random key pre-distribution scheme [11]. In [19] the scheme uses multiple key spaces (numbering τ) and generates with a high probability a common pair-wise key between any two nodes. This enables them to increase the network's resilience to node capture without increasing the memory requirements compared to [15]. While the scheme enhances the resilience of the network against compromised nodes, the resource requirements are still nontrivial. Each node needs to store $\tau(\delta+1)$ entries each equal to the key length. For each communication, a node needs to generate two vectors each of size δ+1, one for the source and the other for the destination and perform a dot product of the two vectors. Furthermore, the key agreement between two nodes that don't share a common space is done through other nodes, which expose it to disclosure if any one of the nodes involved in the key exchange is compromised.

In [17], for each sensor i, the setup server computes a polynomial share of a bivariate t-degree symmetric polynomial $f(x, y)$ computed for node i and hands it to the node. Thus node i is loaded with $f(i, y)$. For any two sensor nodes i and j, node i can compute the polynomial $f(i, j)$ by evaluating $f(i, y)$ at point j. Likewise, node j can compute $f(j, i)$, which is identical to $f(i,j)$ by choice of the polynomial. This serves as the common key between i and j. Again [21] extends this work in a manner motivated by the proposed q-composite extension [2] of the random key pre-distribution scheme [11]. In a following paper [22], the authors integrate location-based knowledge to provide higher probability to establish pair-wise keys between neighbor sensors, better resistance against node captures, and better scalability.

Pietro *e. al.* [23] present an incremental update to random key pre-deployment by considering pseudo-random key deployment based on previous work [18]. This method enhances the channel establishment procedure but adds to the

storage requirement at each sensor. These kinds of protocols are infeasible in situations where a node may communicate with any other node in the network. This is because each time a new destination is considered; the entire key establishment procedure has to be initialized unless there is a large memory to store, in addition to the initial keys and their indices, the transformed keys with all possible destinations.

Du *et al.* [29] present a scheme to use pre-deployment knowledge to improve network connectivity in terms of secure links and resilience against node capture. It was presented to improve the memory requirement compared to [11], but this improvement can benefit any of the key pre-distribution schemes.

We note that all the key pre-distribution schemes provide either no security or probabilistic security against compromised nodes. Probabilistic security assumes thresholds for the number of compromised nodes, beyond which the entire network becomes exposed. The threshold may be exceeded in the event of a localized security breach that affects all the nodes in a geographical region. Our approach, in contrast, provides deterministic security. Compromising any number of nodes is incapable of exposing the communication channel between two uncompromised nodes.

The second flavor of key management protocols is the Kerberos-like protocols [1],[13], and [25]. The idea of using clusters of nodes for key management is suggested by the work on secure Pebble-nets [13]. The authors propose using a single key called the *group key* for group membership and authentication, and another globally shared key called the *Traffic Encryption Key* (TEK) to secure channel communication. A subset of nodes called the backbone nodes has the responsibility of generating and distributing the TEK. The main disadvantage of this work is that it is totally insecure; the compromise of even a single node renders the entire scheme vulnerable. Perrig *et al.* [1] present SPINS, which is based on a master secret key shared between each node and the base station and hash functions to calculate session and *MAC* keys. To establish a secure channel between any two nodes in the network, a shared session key is obtained from the base station. SPINS guarantees data confidentiality, two-party data authentication, and data freshness as long as the base station is not compromised. SPINS uses multiple specialized higher cost base stations with large energy, memory and communication resources to create a tree in the network. Since these base stations are fixed, they are potential targets for security attacks. Compromising, destroying, or jamming a base station used in SPINS renders it impossible to create new secure sessions in the whole section controlled by that base station. Also, if the base station is compromised, the confidentiality of the communication of any node in its group can be destroyed. Since a potentially far-away base station acts as the intermediary for key management, key management in SPINS can be energy inefficient and can lead to high end-to-end delay. Also SPINS does not take into account the possibility of disclosure of the master key by compromising the sensor node. This will result in disclosing all the old communications with this node, if an adversary buffers these communications. It is assumed that session and *MAC* keys are valid throughout the life time of the sensor node, which results in weak security for networks that have a long life time. Since all the node-to-node key agreement is established through the base station, it may result in flooding the base station and exhausting the energy of sensor nodes in the routing path.

Deng *et al.* [25] proposes a protocol for secure data aggregation with base station, sensing node, and aggregators, which act as collectors of data. It establishes mutual trust between a sensor and its assigned aggregator using shared keys. The trust model is used by the sensors to verify the commands of the aggregators and by the aggregators to verify the integrity of the data sent by the sensors. The protocol enables secure communication to and from aggregators but does not solve the general case of secure any-to-any communication between any two nodes.

In general, the proposed Kerberos-like protocols suffer from one or more of the following problems: lack of scalability, high energy over head, high end-to-end delay, and vulnerability to denial of service or compromise targeted at the specialized key management nodes.

There is a large volume of work on secure broadcast or multicast in wireless, and specifically, sensor networks [9],[12],[20], and [26]. The problem addressed there is distinct from our problem definition since they target the secure one-to-many and one-to-all problems, while our focus is one-to-one communication. [30], [31], [32], [33], [34], and [35] present examples of foundational key management protocols that are indirectly related to the key management protocols in sensor networks, presented here for further reading.

7 Conclusions

We have presented the design of a key management protocol called SECOS for resource constrained sensor networks. SECOS divides the sensor field into control groups with a control node in each group. Key exchange between nodes within a control group happens through the mediation of the control node while inter-group communication involves establishing a secure channel between two control nodes with the mediation of the base station. In SECOS, the keys are refreshed and the control nodes changed periodically to ensure higher security. Simulation runs are conducted to bring out the difference in overhead energy expended and data delay between SECOS and SPINS. SECOS is seen to perform better under a wide variety of communication patterns and cache sizes. A security analysis of SECOS is presented and comparison performed with previous protocols. The analysis shows that SECOS can outperform these protocols in terms of the number of compromised nodes that it can tolerate. A mathematical analysis is performed to determine the optimal control group-size in terms of

energy overhead. An upper and a lower bound are derived based on the memory, computational, and bandwidth constraints, the level of security tolerance afforded, and the energy expended in key management.

In the paper, we have addressed the issue of when to trigger the key refreshment and control node change. This involves monitoring anomalous behavior in the network, such as abnormal traffic patterns, which may indicate a security breach. A second issue discussed is the determination of the control node. It is desirable that this be a trusted entity to avoid the energy overhead of changing a control node. We also proposed the use of collaborative monitoring of a sensor node's behavior by its neighbors to determine the trustworthiness of the node.

For future work, we plan to address the problem of choosing the control node according to the availability of resources to perform its privileged function. A control algorithm needs to observe the state of the resources at the nodes in the control group and decide on a schedule for re-selection of a control node. This should itself be a protocol, which is parsimonious in its energy consumption.

References

[1] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, and D.E. Culler, SPINS: Security Protocols for Sensor Networks, Wireless Networks, vol. 8, pp. 521-534, 2002.

[2] H. Chan, A. Perrig, and D. Song, Random Key Predistribution Schemes for Sensor Networks, At the IEEE Symposium on Security and Privacy, pp. 197-213, May 2003.

[3] C. Boyd and A. Mathuria, Key establishment protocols for secure mobile communications: A selective survey, in Australasian Conference on Information Security and Privacy, pages 344–355, 1998.

[4] C. Park, K. Kurosawa, T. Okamoto, and S. Tsujii, On key distribution and authentication in mobile radio networks, in Advances in Cryptology – EuroCrypt '93, pages 461–465, 1993. Lecture Notes in Computer Science Volume 765.

[5] M. Tatebayashi, N. Matsuzaki, and D. B. Jr. Newman, Key distribution protocol for digital mobile communication systems, in Advances in Cryptology – Crypto '89, pages 324–334, 1989. Lecture Notes in Computer Science Volume 435.

[6] Y. W. Law, S. Etalle, and P. Hartel, Key Management with Group-Wise Pre-Deployed Keying and Secret Sharing Pre-Deployed Keying, Technical Report TR-CTIT-02-20, Department of Computer Science, University of Twente, July 2002.

[7] Y.W. Law, R. Corin, S. Etalle, and P.H. Hartel, A Formally Verified Decentralized Key Management Architecture for Wireless Sensor Networks, Personal Wireless Communications (PWC 2003), Sep 2003. Lecture Notes of Computer Science, Volume 2775, Springer-Verlag.

[8] R. Gennaro and P. Rohatgi, How to sign digital streams, in Cryptology – Crypto'97, Lecture Notes in Computer Science, Vol. 1294, pp. 180-197.

[9] A. Perrig, R. Canetti, J. Tygar, and D. Song, Efficient authentication and Signing of multicast streams over lossy channels, in IEEE Symposium on Security and Privacy, 2000.

[10] P. Rohatgi, A compact and fast hybrid signature scheme for multicast packet authentication, in ACM Conference on Computer and Communications Security, 1999.

[11] L. Eschenauer and V.D. Gligor, A key management scheme for distributed sensor networks, in Proceedings of the 9th ACM Conference on Computer and Communication Security, pages 41–47, November 2002.

[12] D. Liu and P. Ning, Efficient Distribution of Key Chain Commitments for Broadcast Authentication in Distributed Sensor Networks, in Proceedings of the 10th Annual Network and Distributed System Security Symposium, pages 263--276, February 2003.

[13] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti, Secure pebblenets, in Proceedings of the 2001 ACM Intl. Symp. on Mobile Ad Hoc Networking and Computing (Mobihoc '01), pages 156-163. ACM Press, October 2001.

[14] Bruce Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons, 2nd edition.

[15] R. Blom, An optimal class of symmetric key generation systems, Advances in Cryptology: Proceedings of EUROCRYPT 84 (Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, eds.), Lecture Notes in Computer Science, Springer-Verlag, pp. 209-335 and 338, 1985.

[16] W. Stallings, Cryptography and Network Security: Principles and Practices, third edition, Prentice Hall, 2003.

[17] 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, pages 471-486, 1993.

[18] S. Zhu, S. Setia, and S. Jajodia, A distributed group key management protocol for ad hoc networks, Unpublished manuscript, December 2002, George Mason University, VA-USA.

[19] W. Du, J. Deng, Y. Han, and P. Varshney, A Pair-wise Key Pre-distribution Scheme for Wireless Sensor Networks, in Proceedings of the 10th ACM conference on Computer and communication security (CCS'03), Washington D.C., USA. October 27-30, 2003.

[20] L. Lazos and R. Poovendran, Energy-aware secure multicast communication in ad-hoc networks using geographical location information, ICASSP 2003, Hong Kong, China, April 2003.

[21] D. Liu and P Ning, Establishing Pair-wise Keys in Distributed Sensor Networks, in Proceedings of the 10th ACM conference on Computer and communication security (CCS'03), Washington D.C., USA. October 27-30, 2003.

[22] D. Liu and P Ning, Location Based Key Establishment for Static Sensor Networks, in ACM Workshop of Ad hoc and Sensor networks (SASN'03).

[23] R. Pietro, L. Mancini, and A. Mei, Random Key Assignment for Secure Wireless Sensor Networks, in ACM Workshop of Ad hoc and Sensor networks (SASN'03).

[24] S. Zhu, S. Setia, and S. Jajodia, LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks, in Proceedings of the 10th ACM conference on Computer and communication security (CCS'03), Washington D.C., USA. October 27-30, 2003.

[25] J. Deng, R. Han, and S. Mishra, Security Support for In-Network Processing in Wireless Sensor Networks, in ACM Workshop of Ad hoc and Sensor networks (SASN'03).

[26] D. Bruschi and E. Rosti, Secure multicast in wireless networks of mobile hosts: protocols and issues, ACM/Baltzer Mobile networks and applications, special issue on multipoint communication in Wireless Mobile Networks, Vol. 6, No. 7, December 2002.

[27] J. Deng, R Han, and S. Mishra, The Performance Evaluation of Intrusion-Tolerant Routing in Wireless Sensor Networks, in Proc. of IEEE 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03), LNCS 2634.

[28] S. Zhu, S. Xu, S. Setia, and S. Jajodia, Establishing Pair-wise Keys For Secure Communication in Ad Hoc Networks: A Probabilistic Approach, in the 11th IEEE International Conference on Network protocols (ICNP'03), Atlanta, Georgia, November 4-7, 2003.

[29] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney, A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge, in IEEE INFOCOM'04, March 7-11, 2004, Hong Kong.

[30] B. C. Neuman and T. Tso, Kerberos: An authentication service for computer networks, IEEE Communications, vol. 32, no. 9, pp. 33–38, September 1994.

[31] M. Tatebayashi, N. Matsuzaki, and D.B.J. Newman, Key distribution protocol for digital mobile communication systems, Advances in Cryptology-CRYPTO'89, LNCS Volume 435, pp. 324–334, 1989, Springer-Verlag.

[32] C. Park, K. Kurosawa, T. Okamoto, and S. Tsujii, On key distribution and authentication in mobile radio networks, Advances in Cryptology-EuroCrypt'93, LNCS Volume 765, pp. 461–465, 1993, Springer-Verlag.

[33] M. Beller and Y. Yacobi, Fully-fledged two-way public key authentication and key agreement for low-cost terminals, Electronics Letters, vol. 29, no. 11, pp. 999–1001, 1993.

[34] D. Wong and A. Chan, Efficient and mutually authenticated key exchange for low power computing devices, in Proc. ASIACRYPT, December 2001.

[35] A. D. Wood and J. A. Stankovic, Denial of service in sensor networks, IEEE Computer, 35(10):54–62, October 2002.

[36] National Bureau of Standards (NBS), Specification for the data encryption standard, Federal Information processing Standards (FIPS) Publication 46, 1977.

[37] J. Daemen and V. Rijmen, AES proposal: Rijndael, 1999.

[38] D. Wheeler and R. Needham, TEA, a Tiny Encryption Algorithm, 1994. http://www.ftp.cl.cam.ac.uk/ftp/papers/djw-rmn/djw-rmn-tea.html.

[39] R. L. Rivest, The RC5 encryption algorithm, in Workshop on Fast Software Encryption, pp. 86-96, 1995.

[40] S. Noel, D. Wijesekera, C. Youman, Modern Intrusion Detection, Data Mining, and Degrees of Attack Guilt, in Applications of Data Mining in Computer Security, Daniel Barbarà and Sushil Jajodia (eds.), Kluwer, 2002.

[41] S. Banerjee and S. Khuller, A Clustering Scheme for Hierarchical Control in Multi-hop Wireless Networks, in *Proceedings of IEEE INFOCOM*, April 2001.

[42] S. Bandyopadhyay and E. Coyle, An Energy-Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks, in *Proceedings of IEEE INFOCOM*, April 2003.

[43] V. Kawadia and P. R. Kumar, Power Control and Clustering in Ad Hoc Networks, in *Proceedings of IEEE INFOCOM*, April 2003.

[44] B. McDonald and T. Znati, Design and Performance of a Distributed Dynamic Clustering Algorithm for Ad-Hoc Networks, in *Annual Simulation Symposium*, 2001.

[45] O. Younis and S. Fahmy, Distributed Clustering for Scalable, Long-Lived Sensor Networks, *Purdue University, Technical Report CSD TR-03-026*, June 2003.

[46] S. Marti, T. J. Giuli, K. Lai, and M. Baker, Mitigating routing misbehavior in mobile ad hoc networks, ACM/IEEE International Conference on Mobile Computing and Networking, 2000.

[47] M. Krasniewski, P. Varadharajan, B. Rabeler, S. Bagchi, and Y. C. Hu, TIBFIT: Trust Index Based Fault Tolerance for Arbitrary Data Faults in Sensor Networks, accepted to appear in the International Conference on Dependable Systems and Networks (DSN '05), Yokohama, Japan, June 28 - July 1, 2005.

[48] D. Estrin, Mani Srivastava, and Akbar Sayeed, Wireless Sensors Networks, MobiCOM 2002 Tutorial no. 5. Available at: http://nesl.ee.ucla.edu/tutorials/mobicom02.

[49] H. Chan and A. Perrig, PIKE: Peer Intermediaries for Key Establishment in Sensor Networks, *IEEE INFOCOM*, 2005.

[50] F. Ye, H. Luo, S. Lu, and L. Zhang, Statistical En-route Detection and Filtering of Injected False Data in Sensor Networks, *IEEE INFOCOM* 2004.

[51] TinyOS. http://www.tinyos.net and http://www.xbow.com.

[52] A. Woo and D. Culler. A Transmission Control Scheme for Media Access in Sensor Networks. MOBICOM 2001.

[53] http://www.cerias.purdue.edu/homes/crisn/courses/cs555/cs555_lect5.pdf

[54] J. Newsome, E. Shi, D. Song, and A. Perrig, The Sybil attack in Sensor Networks: Analysis & Defenses, IPSN 2004, pp. 259-268.

[55] C. Karlof and D. Wagner, Secure Routing in Sensor Networks: Attacks and Countermeasures, SNPA 2003.

[56] I. Khalil, S. Bagchi, and N. Shroff, LiteWorp: A Lightweight Countermeasure for the Wormhole Attack in Multihop Wireless Networks, accepted to appear in the International Conference on Dependable Systems and Networks (DSN '05), Yokohama, Japan, June 28 - July 1,2005, available at:

http://shay.ecn.purdue.edu/~dcsl/Publications/papers/93_Khalil_I_final.pdf.

[57] I. Khalil, S. Bagchi, and C. Nina-Rotaru, DICAS: Detection, Diagnosis and Isolation of Control Attacks in Sensor Networks, accepted to appear at IEEE/CreateNet SecureComm 2005, Athens, Greece, 5th-9th September, 2005, available at: http://shay.ecn.purdue.edu/~dcsl/Publications/papers/khalil_DICAS.pdf.

[58] G. Jolly, M. C. Kusçu, P. Kokate, and M. F. Younis, A Low-Energy Key Management Protocol for Wireless Sensor Networks, ISCC 2003: 335-340.

Appendix

[I] Timers and Threshold Values

The following table presents a summary of the timers and the threshold values used in SECOS.

	Name	Description	Tradeoffs
1	Session & authentication key refreshment timer	When the timer expires, the session and authentication keys are refreshed applying a *MAC* function on the *SC(M,S)* XOR-ed with the volatile secret key and concatenated with 1 for the session key and 2 for the authentication key.	A higher value makes it less secure by facilitating cryptanalysis and allowing past communication of a compromised node to be divulged. A lower value makes it energy inefficient.
2	Control node refreshment timer (T_{ctrl})	When the timer expires the control node is changed. A new control node is selected and delivered the list of control group members. The old control node returns to the normal sensing mode.	A higher value makes it less secure in case the control node gets compromised. A lower value makes it energy inefficient.
3	Opinion counter threshold value ($T_{counter\_threshold}$)	When the opinion counter at a node, *X*, crosses the threshold for a certain monitored node, *Y*, then *X* sends the opinion counter value and the ID of *Y* to the base station	A higher value makes it less secure since many malicious events may not be detected because they do not increment the opinion counter to the threshold value. A lower value makes it energy inefficient.
4	Alert collection timer ($T_{suspect\_collection}$)	When the timer fires, the base station either starts correlating the received alerts if they are sufficient, or polls certain nodes to send their opinion counters to collect sufficient alerts.	A higher value allows sufficient alerts from most involved observer nodes to arrive to the base station. But it makes the network less secure by delaying the malicious event detection and response.
5	Trust level threshold (T_{trust_level})	When the trust level of a node, *X*, in the network goes below the threshold, the base station declares *X* as a malicious node.	A higher value makes it more secure since only highly trusted nodes are allowed in the network. But it may result in high node revocation due to false alarms by natural faults and communication errors.

Table 3: Timers and Threshold Values in SECOS

[II] Notations

This section provides a summary of the notations used throughout the paper.

Acronym	Description	Acronym	Description
S	A generic sensor node	*C*	A generic control node
M	The base station	*N*	The total number of nodes in the network
D	The density of the nodes in the network	*R*	The communication range
MAC	Message Authentication Code	*E(K,X)*	Encryption of message *X* using key *K*
MAC(*K*,*Z* ⊕*X*||*Y*)	The application of the *MAC* algorithm, keyed by key *K*, to the result of the concatenation of *Y* with the result of *Z* XOR-ed with *X*	*H(X)*	The hash value of the message *X*

MK_{AB}	The master key shared between *A* and *B*	VK_{AB}	The volatile secret key shared by *A* and *B*
SK_{AB}	The session (encryption/decryption) key shared between *A* and *B*	AK_{AB}	The Authentication (*MAC*) key shared between *A* and *B*
RK_{AB}	The random number generator key shared between *A* and *B*	K_{AB} (=K_{BA})	Any secret key (MK_{AB},VK_{AB}, SK_{AB}, AK_{AB}, RK_{AB}) shared between *A* and *B*
SG_{ctrl}	The size of the control group (*i.e.*, the number of nodes in the control group)	*S(Pkt)*	The size of the *Pkt* packet. *Pkt* is one of the packets defined in Table 1
SG_{com}	The communication group size	S_R	Size of the key reply (*i.e.*, S_R=*S(K_rep)*)
S_{Key}	The amount of storage required to store a cryptographic key such as the session key	H_{ctrl}	The average number of hops between a pair of nodes in a control group
H_{com}	The average number of hops between a pair of nodes in the communication group	H_{all}	The average number of hops between a pair of nodes in the whole network
N_{BC}	The number of control groups within one communication group	N_G	The number good (uncompromised) nodes in the network
N_C	The number of compromised nodes in the network	N_B	The number of control groups in the network
MalC(*i*,*j*)	The malicious counter at node *i* about node *j*	$MalC_{max}$	Maximum value of the malicious counter
$N_m(i)$	The number of monitors of node *i* that report their opinions to the base station	$T_{counter\_threshold}$	The threshold value of the malicious counter above which a node becomes suspicious
$L_{assurance}$	The level of detection assurance at a monitoring node a bout a suspected event	$L_{trust}(i)$	The trust level of node *i* that is maintained by the base station
T_{trust_level}	The trust level threshold beyond which the base station identify a node as malicious	*Sync_diff*	The maximum acceptable difference between the counters shared by a pair of nodes in the network
$T_{suspect\_collection}$	The time the base station waits to collect more opinions a bout a suspected event starting from time of the first arrived opinion	*SC(i,j)*	The sending counter value of node *i* that is shared with node *j* (*SC(i,j) = RC(j,i)*)
RC(i,j)	The receiving counter of node *i* that is shared with node *j* (*RC(i,j) = SC(j,i)*)	$Counetr_{ij}$	Refers to both *SC(i,j)* and *RC(i,j)*
T_{Comp}	The time that is minimally required to compromise a node	E_1	The event that the control node of a certain control group is compromised
E_2	Thee event that there is at least one compromised node in the bounding path between a pair of nodes in the control group	E_3	The event that the control node lies in the bounding path between a pair of nodes in the same control group
$P_{C(A-B)}$	The probability of compromising the link between *A* and *B*	N_{bp}	The number of nodes within the bounding path between a pair of nodes in the same control group
P_{Lerr}	The probability of natural error in a packet over a link between a pair of neighbor nodes	P_{CD}	The probability that a node is compromised and dropping packets
S_C	The regular cache size at each node	S_{CC}	The control cache size at each node
α_C	The hit rate in the regular cache (*i.e.,* the probability of finding an element in the cache)	β_C	The miss rate in the regular cache (*i.e.,* the probability of not finding an element in the cache, β_C =1-α_C)
α_{CC}	The hit rate in the control cache (*i.e.,* the probability of finding an element in the cache)	β_{CC}	The miss rate in the control cache (*i.e.,* the probability of not finding an element in the cache, β_{CC} =1-α_{CC})
T_{ctrl}	The average time a node stays in the control role for a single round	E_{nergy}	The energy for the transmission and the reception of a single bit
G_{COMP}	The maximum control group size under the computational limitation only	G_{BW}	The maximum control group size under the bandwidth limitation only

G_{SEC}	The maximum control group size under an acceptable number of compromised sessions.	G_{STORE}	The maximum control group size under the storage limitation only
μ	The reciprocal of the rate of the Poisson process used for changing the destination of a packet (*i.e.,* a new destination is selected on average every μ time units)	λ	The reciprocal of the rate of the Poisson process used for data packet generation (*i.e.,* one packet is generated on average every λ time units)
BW	The channel bandwidth	N_{nbr}	The average number of one hope neighbors of a node
T_E	The total overhead energy		

[III] Message Overhead

In this section, we analyze the overhead in terms of control messages for each of the operations in SECOS. The overhead is calculated as the product of the number of bytes and the number of hops.

Some Notation: Let N_{nbr} be the average number of neighbors of a node, H_{cmax} be the maximum number of hops between any two nodes in the control group, and D be the density of nodes in the network. Further, R is the range of transmission, and H_{com}, H_{ctrl}, and H_{all} are the average number of hops between nodes within the same communication group, between a node and the control node, and between a node and the base station, respectively.

We now calculate the overhead involved in the various functions of SECOS

1. Building the neighbor list: (i) One HELLO message from a node to its neighbors, (ii) N_{nbr} HELLO reply messages from the neighbors to the node, and (iii) one message containing the list of neighbors from the node to the base station. The size of each HELLO or the HELLO reply message is 9 bytes; 8 for the IDs of the sender and the receiver, and one holding the packet data. The size of the neighbor list packet is $4(N_{nbr}+2)$ bytes. The HELLO message travels one hop where the neighbor list message travels H_{all} hops on average to the base station. The total overhead in byte-hop product equals $9(N_{nbr}+1) + 4(N_{nbr}+2)H_{all}$.

2. Setting the control node: (i) One message holding the list of members of the control group from the base station to the control node, (ii) one message for control announcement from the control node to the members of control group, and (iii) one message for neighbor list announcement from the control node to its neighbors. The member list message travels H_{all} hops on average and its size equal to $12\times SG_{ctrl}$ bytes; 4 bytes for each member node ID and 8 bytes for the session key between the member and the control node. The size of the control announcement is 5 bytes and it travels H_{cmax} hops. The number of nodes involved in broadcasting the announcement depends on the range of transmission R and density of nodes in the network D. This number equals to $\pi\times(R\times H_{cmax})^2 D$. The size of the neighbor list is $4 N_{nbr}$ and it travels one hop. The total overhead in byte-hop product equals $12\times SG_{ctrl}\times H_{all} + 5\pi (R\times H_{cmax})^2 D + 4 (N_{nbr}+1)$.

3. Key establishment within the same control group: (i) One message holding the key from the initiator to the target, (ii) one message holding the *Envelop* from the initiator to the control node, and (iii) one message holding the *Envelop* from the control node to the target. The message holding the key travels H_{ctrl} hops on average and its size equals to 16 bytes, 8 bytes for the ID's of the initiator and the target and 8 bytes for the key. The message holding the *Envelop* also travels H_{ctrl} hops on average and its size equals 44 bytes, 8 bytes for the ID's of the initiator and the target of the communication, 8 bytes for the ID's of the intermediate sender and receiver of the message, 8 bytes for the key, 10 bytes for the hash value of the key, and 10 bytes for the *MAC* value, which provides freshness to the message. The total overhead in byte-hop product equals $104\times H_{ctrl}$.

4. Key establishment across control groups with a shared key already exists between the corresponding control nodes: (i) One message holding the key from the initiator to the target, (ii) one message holding the *Envelop* from the initiator to its control node, (iii) one message holding the *Envelop* from the control node of the initiator to the control node of the target, (iv) one message holding the *Envelop* from the target's control node to the target. Message (i) travels H_{com} hops on average and its size equals to 16 bytes, 8 bytes for the ID's of the initiator and the target and 8 bytes for the key. Message (ii) or message (iv) travels H_{ctrl} hops on average and its size equals to 44 bytes, 8 bytes for the ID's of the initiator and the target, 8 bytes for the ID's of the intermediate sender and receiver of the message, 8 bytes for the key, 10 bytes for the hash value of the key, and 10 bytes for the *MAC* value, which provides freshness to the message. Message (iii) travels H_{com} hops on average and its size equals to 44 bytes, 8 bytes for the ID's of the initiator and the target, 8 bytes for the ID's of the intermediate sender and receiver of the message, 8 bytes for the key, 10 bytes for the hash value of the key, and 10 bytes for the *MAC* value, which provides freshness to the message. The total overhead in byte-hop product equals $60\times H_{com} + 88\times H_{ctrl}$.

5. Key establishment across control groups with no shared key between the corresponding control nodes: The same messages as in the previous case in addition to (i) one message holding a key from the initiator's control node to the base station and (ii) one message holding the same key from the base station to the target's control node. The size of each of these messages equals to 16 bytes, 8 bytes for the ID's of the initiator and the target and 8 bytes for the key, each of them travels H_{all} hops. The total overhead in byte-hop product equals $32 \times H_{all}$.

6. Neighbor watch and control node monitoring: One message from a sensor to the base station holding the opinion counter. The size of the message is 13 bytes; 8 bytes for the IDs of the sender and the base station, 4 bytes for the ID of the monitored node, and one byte for the counter. The message travels H_{all} hops on average. The total overhead in byte-hop product equals to $9 \times H_{all}$. This is the overhead when a suspicious node is detected.

Issa Khalil received the B.Sc. degree in computer engineering from Jordan University of Science and Technology (JUST), Jordan, in 1994, and the MS degree in computer engineering from JUST in 1996. He is currently pursuing a PhD in the Dependable Computing Systems Lab of Prof. Bagchi S. His research interest includes key-management, secure routing protocols, and intrusion detection in Ad Hoc and Sensor networks. He has worked as the director of computer and research center of Alquds Open University, West Bank, for more than 6 years.

Saurabh Bagchi joined the department of Electrical and Computer Engineering at Purdue University in West Lafayette, Indiana as an Assistant Professor in August 2002. Before that, he did his Ph.D. from the Computer Science department of the University of Illinois at Urbana-Champaign with Prof. Ravishankar Iyer at the Coordinated Science Laboratory. His Ph.D. dissertation was on error detection protocols in distributed systems and was implemented in a fault-tolerant middleware system called Chameleon.

Ness B. Shroff received his Ph.D. degree from Columbia University, NY in 1994. He is currently a full Professor in the School of Electrical and Computer Engineering at Purdue University. His research interests span the areas of wireless and wire line communication networks, and more recently network security. Dr. Shroff is an editor for IEEE/ACM Trans. on Networking and the Computer Networks Journal, and past editor of IEEE Communications Letters. He was the conference chair for the 14th Annual IEEE Computer Communications Workshop in Estes Park, CO, October 1999) and program co-chair for the symposium on high-speed networks, Globecom 2001 (San Francisco, CA, November 2000). He was the Technical Program co-chair for IEEE INFOCOM'03 and panel co-chair for ACM Mobicom'02. He received the NSF CAREER award in 1996 and the best paper of the year award for Computer Networks, 2003.