SECOS: Key Management for Scalable and Energy Efficient Crypto On Sensors

Issa Khalil, Saurabh Bagchi
Dependable Computing Systems Lab
School of Electrical & Computer Engineering, Purdue University
465 Northwestern Avenue, West Lafayette, IN 47907.
Email: {ikhalil,sbagchi}@purdue.edu

Abstract

Wireless sensor networks are becoming a critical computational infrastructure, in which the communication between nodes needs to be protected from eavesdropping and tampering. Symmetric key cryptography is the fundamental technique being used. The protocols in this domain 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 this paper, we propose a protocol called SECOS that mitigates these problems. SECOS divides the sensor field into control groups each with a control head. Data exchange between nodes within a control group happens through the mediation of the control head which provides the common key. The keys and the control heads are changed periodically to enhance security. SECOS enhances the survivability of the network by handling failures of control nodes. The experiments based on a simulation model show 7 times reduction in energy overhead and 50% reduction in latency compared to the state-of-the-art protocol, SPINS. We also provide an analytical derivation of the optimal control group size that operates under the resource constraints and minimizes energy consumption.

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

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 the 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

the inter-node communication protected against tampering for protection against possible terrorist attacks against critical civilian infrastructures.

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 the 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 division and modulus operations by a large number. The common approach 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 of some nodes being 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. 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 very 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 which 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 a protocol called SECOS (**S**calable & **E**nergy-Efficient **C**rypto **O**n **S**ensors) for key management in sensor networks that use symmetric cryptography. SECOS provides primitives for secure any-to-any communication in a large scale sensor network. It minimizes the fallout of compromising some nodes in the network. It incorporates a protocol for key distribution which is energy efficient and reduces the demands on the constrained bandwidth and computational resources of the sensor nodes. 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 generated by the control node, while inter-group communication involves establishment of 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 not have any resource constraints, while all the rest, including the control nodes, are generic sensor nodes. A key principle in SECOS is key refreshment, whereby new keys are generated using one-way hash functions on the old keys. A second principle is to change the nodes which play a privileged role in the key management. A third principle is to use key caches consistently at all the nodes to minimize the energy consumption. However, the caches are purged periodically either on a time schedule or in response to some events, such as control node change. The principles are designed to mitigate the effect of a node being compromised.

A key decision choice in SECOS is the control group size. We present a mathematical analysis to determine the upper bound on the control group size, due to the resource constraints on the control node and the allowable security quantified by the number of messages that are exposed due to nodes being compromised. We then present an equation that quantifies the energy cost of key management in terms of several factors, including the control group size and the communication group size. 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 1.05-1.50. A large cache means keys are available locally and then SECOS performs comparably to SPINS.

However, it also implies weaker security to compromised nodes. With respect to variation of the control group size, the maximum benefit to SECOS is observed when the control group size becomes equal to the communication group size.

To the best of our knowledge, this is the first work that addresses the problem of key distribution in sensor networks accounting for the combination of issues – security, resource constraints, low cost, and scalability. The paper makes the following important contributions.

1. It provides protocols for key distribution and revocation that are sensitive to the sensor node's resource constraints, including computation, communication, and bandwidth.
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. It optimizes the adverse impact of some nodes being compromised due to attacks, or being unavailable due to natural failures.

The rest of the paper is organized as follows. Section 2 refers to related research. Section 3 presents the design of SECOS. Section 4 provides a mathematical analysis of the control group size and energy consumption. Section 5 describes the experiments and the results. Section 6 concludes the paper with mention of future work.

2 Related Research

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], and [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, such as 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],[12][14]). These protocols 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. Others consider probabilistic key pre-distribution techniques, but they focus on group and broadcast communication [8]. 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. For example, the work in [9] requires over 1Kbyte of authentication information per packet and the extension in [11] still requires 300 bytes per packet. The signature length that they use to authenticate the packets is $D.(m+log_2m)$, where D is the number of bits required to represent the elements in the domain of the one-way hash function used in their signing algorithm and m is the length of the message in bits.

Several researchers have tried to address the problems of classical symmetric protocols in the context of sensor networks. In [12], the authors consider a random key pre-distribution that uses a large pool of keys from which they poll m keys at random which are loaded into each sensor node before deployment. When a node is deployed, it initiates the shared-key discovery phase. Then the node uses its established secure channels with other nodes to build secure channels with its neighboring nodes that it has no common keys with. However, compromising any node reveals all the keys in the ring. The key establishment process is open to compromise since the identifiers are broadcast to a receiver set that has not been authenticated. The authors in [2] improve on this work by requiring more than one key to be shared between any two nodes to establish a secure communication. It also uses partial key exchanges on multiple paths to ensure security from some nodes on the path being compromised. But it adds substantial overhead in finding multiple disjoint paths and storing multiple keys.

TESLA [10] is a protocol for authentication of a broadcast source in lossy channels. TESLA offers sender authentication, strong loss robustness, high scalability, and minimal overhead, at the cost of loose initial time synchronization and slightly delayed authentication. However, it has too high a memory and communication overhead to make it suitable for resource starved sensor nodes. In [1], the authors improve TESLA with a version called μTESLA that uses symmetric keys for key chain commitment and limits the number of key disclosures to one per time interval instead of one per packet. μTESLA uses the base station to channel broadcasts from any sensor node. The work called SPINS, also presents SNEP 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. But this work 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 communication with this node. 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 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. In μTESLA, the key distribution is achieved by direct one-to-one communication that is not scalable for large networks. The work in [13] addresses this problem which minimizes the overhead for key commitment distribution but adds a lot of contention on the storage capabilities on the sensor nodes.

Our idea of using clusters of nodes for key management is suggested by the work on secure Pebble-nets [14]. 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 the compromise of even a single node renders the entire scheme vulnerable.

3 Description of SECOS

Our high-level design goals in SECOS are to (i) Provide a secure key distribution channel. (ii) Minimize the adverse fallout of compromising any sensor node. (iii) Make key management energy efficient. (iv) Reduce the end-to-end delay of data communication.

We follow a few underlying principles in our protocol design.

1. *Refreshing the keys*. The keys are periodically refreshed to guard against compromised nodes using old keys to eavesdrop on communication. The keys are refreshed by one way hash functions or MAC functions.
2. *Changing the nodes which play a privileged role*. A node which has a privileged role in key distribution is a crucial part of the key management infrastructure. 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 privileged nodes being compromised and provide for them to be changed on a time schedule, or when triggered by anomalous events.

3. *Encash the caches but judiciously*. Key caches can give enormous energy savings, but can also be a source of vulnerability in the event of a node with a large cache of stored keys being compromised. Therefore, we purge caches regularly, e.g., when the node's role changes, or when some privileged node reorganizes the network.

3.1 Keys in SECOS

SECOS uses an initial secret key that is burnt into each sensor node at manufacture time. This key is shared between the node and the base station. The initial key, called the initial master key, 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 for channel encryption.

The base station has a series of MAC functions that it uses for key generation. Each sensing node also has the same set of MAC functions. The base station also shares two counters with each sensing node, one for each direction of communication. Let us call them the *master2nodecounter* and *node2mastercounter* respectively. These counters are kept synchronized by incrementing them on message sent or received between the sensor and the base station. The counters have a dual role to play. First, they are used for semantic security to prevent the replay of messages by an adversary. Second they are used for session key generation. Initially the base station generates a session key for each node using a MAC function (say, MAC1) on the master2nodecounter keyed by the master key. The same session key can be generated by the node since it has the same MAC function, the master key and the counter. A similar mechanism is used to generate a shared authentication key between the base station and the sensing node. In order to achieve higher security, SECOS uses key refreshment for the session key and the authentication key.

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. There are four kinds of keys – the master key, the session key, the MAC key, and the random number generator key which will be denoted respectively as $K_{AB(1)}$, $K_{AB(2)}$, $K_{AB(3)}$, and $K_{AB(4)}$. $K_{AB}(X)$ denotes the encryption of a message X using key K_{AB}. $MAC(K, X|Y)$ is the hash of message X concatenated with Y, using key K. 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.

3.2 Hierarchical Structure

We will use an initial flat design to motivate the hierarchical structure present in SECOS. In the flat layout, there are two categories of nodes – a powerful base station and sensing nodes. Each sensing node shares with the base station a master key, two counters, and MAC functions. The base station is solely responsible for key management in the network and acts as the intermediary for any communication. When a node A wishes to initiate communication with a node B, it requests the base station for a shared session key. The base station generates the key and sends it to both A and B using the secure channel that it has with each. This key is only valid till the time that the shared key between the base station and the individual node is valid. Each node has a cache in which it may store the shared session key.

This flat layout is good for small networks but does not scale well with network size since the nodes in the neighborhood of the base station are flooded by key requests and replies. Also in large networks the average number of hops to the base station increases which means the energy consumed for key requests and replies increases drastically. Due to larger distances, the end-to-end data latency also increases.

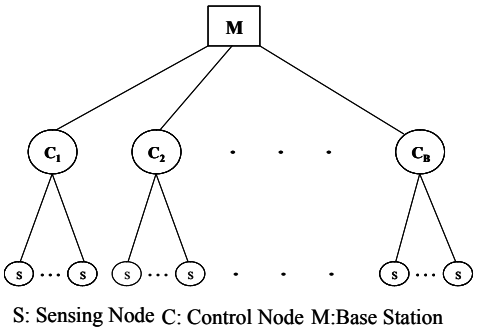


Figure 1: Two level hierarchy for key management in SECOS

This motivates the need for having a hierarchical structure of nodes in the sensor network. The hierarchical structure we propose has clusters of sensor nodes based on geographical proximity. Each cluster has a specially designated node called the *Control Node* and the cluster is called a *Control Group*. We do not impose any special requirements on the control node, and it can be an ordinary sensing 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 in place of the base station in the flat layout. It is a trusted node which is earmarked by the base station. It will periodically be changed for the purpose of security (the previous control node being compromised), and for more even energy drain (the control node drains its energy faster).

Each sensor node has 2 types of caches: (i) *Regular cache:* It 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 sends a key request. Subsequently, it may generate more data packets for the same receiver, before the key request has been satisfied. So, this cache stores the receivers for which there is an outstanding key request.

3.2.1 Creating & Changing Control Node

The base station designates a node as a control node for a group, say *C*, and sends it a list of session keys for each node in the group. The session key is not sent to the sensing node in the group. The sensing node generates the key on its own as it did before for the session key shared between the node and the base station, i.e. it uses a MAC function keyed by the master key applied to the shared counter value. The control node C does not perform any sensing work and uses all its available storage to store these keys. After C receives the list, it broadcasts to the group members a message claiming it is the new control node for the group. When a group member receives the claim, it challenges C. The heart of the challenge lies in generating a random number, encrypting it with the session key that should be available at the legitimate control node, asking C to do some processing on the number and send it back encrypted. The details are provided in the **Appendix.** Note that now the node has a shared session key with the control node, which is separate from the shared session key with the base station. The initial control node set up is shown in Figure 2(a).

The control node is changed by the base station based on a certain time schedule, or when some anomalous events happen, e.g., the trust level of the control node changes. When the base station decides to initiate the change, it first selects a new control node and hands it a *new* set of session keys for the group members. The new control node broadcasts its presence to the group. In response, 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.

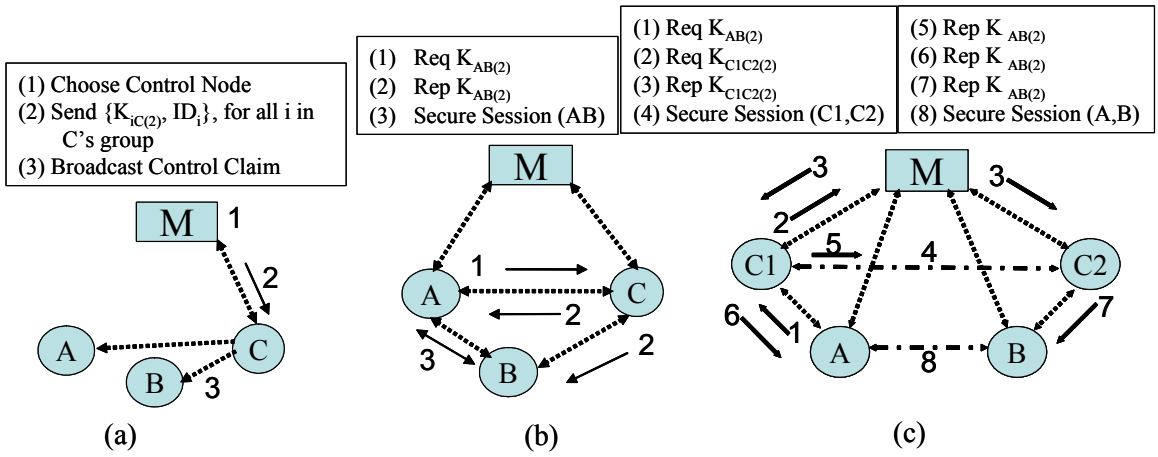
3.2.2 Node to node communication within control group

When a node, say A, needs to communicate with another node, say B, it first checks in its regular cache for the session key between itself and B. If present, it uses this. If not present, it contacts the control node. The control node checks if B lies in the same control group. If it does, it generates a new session key for A and B and

passes it through the secure channel to both A and B. 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 2(b).

3.2.3 Node to node communication across control group

If the destination A wishes to communicate with a node that lies in a different control group, then two control nodes will have to be involved. Say A lies in group G1 and B in G2 and the respective control nodes are CN1 and CN2. CN1 checks its control cache for the session key between itself and CN2. If not present, it asks the base station to generate a session key between CN1 and CN2. This is generated and sent to CN1 and CN2 using the secure channel between the base station and the individual control nodes. CN1 now generates a shared session key between A and B and passes it down to A. It passes it to CN2 which forwards it to B. Now a secure communication channel is established between A and B. The inter-group communication is shown schematically in Figure 2(c).



Req: Key request; Rep: Key reply; $K_{xy(2)}$: Session key between X & Y
M: Base Station; A,B: Sensing Nodes; C: Control Node

Figure 2: (a) New control node setup; (b) Intra-group communication using the control node; (c) Inter-group communication using two control nodes. The two control nodes do not have a secure session when the process starts.

In addition to the two types of cache that a sensing node has, 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. The size of this cache is equal to the size of the control group. (ii) *Control cache:* It stores the session keys the node uses to communicate with other control nodes for inter-group communication.

3.2.4 Protecting the Control Node

SECOS introduces control nodes as the new infrastructure entities for key management. Therefore, a concern may be its capacity to handle attacks directed at the control nodes or tolerating their unavailability due to failures. Here we qualitatively discuss the system defenses against these.

1. *Impersonating the control node.* This is not possible since all the communications between the control node and all other nodes in its control group are protected by encryption and authentication, and the sensors challenge the node that claims to be the control node. The only way of doing so is to compromise the control node to get the keys in its ring.
2. *Compromising the control node.* The probability of compromising the node is very slim, since these control nodes are not fixed, and the frequency of change of the control node can be set to account for the possibility of compromise. In future work, we plan to address the detection of compromised nodes. When a new control node becomes active, the old control node purges all its contents; this prevents the attacks directed to the old control node from getting any useful information.
3. *DOS attack directed to the control node.* This may be launched through a node repeatedly asking for a key. This kind of attack is handled by keeping state at the control node for the current active nodes (nodes that recently asked for keys), which is a sliding-window based state, 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. Another defense against this attack is to allow the other party, which receives a key reply in response to the compromised node's key request, to send a feedback if the initiator does not establish the session within a certain time threshold.
4. *Unavailability of control node due to failure.* When a node in the control group cannot contact the control node, it sends a key request directly to the base station. The base station verifies the request is coming from a legitimate group member and if it finds the control node is non-existent, installs a new control node.

3.2.5 Key Refreshment

Key refreshment is an important technique in SECOS used to reduce the adverse fallout of nodes being compromised. The four keys that are available in each of the sensors (master key, session key, authentication

key, and random number generator key) are refreshed periodically or when triggered by a certain event. The event may be an anomalous event, such as the detection of an attack, or a natural event, such as low remaining battery in the control node. Since the keys are symmetric the refreshment is done synchronously by the two end-points. The master key refresh is initiated by the base station and is achieved by applying a one way hash function, called F_1, to the old master key. It is applied as follows: $K_{MA\_NEW} = F(K_{MA\_OLD})$. The encryption and authentication keys are refreshed by applying MAC functions, called F_2 and F_3 respectively, keyed by the current master key to a random value sent by the base station when it initiates the refreshment. For example, to refresh the session key between the base station M and a sensing node S, $K_{MS(2)\_NEW} = F_2(K_{MS(1)}, \textit{randval})$. The random value is generated by using the MAC function F_4 keyed by the random number generator key and using the counter shared between the base station and the sensing node. The mathematical formulations for the different key refreshments are included in the appendix.

3.3 Comparison with SPINS

The SPINS protocol described in [1] serves as the starting point for SECOS. SPINS has in common with SECOS the notions of a master key shared between a node and a base station, data authentication using MAC keys that are generated from the master key, and semantic security and replay protection using shared counters. However, SECOS is an alternate multi-level architecture that addresses some of the security and energy consumption concerns of SPINS.

1. SPINS uses multiple specialized higher cost base stations with large energy, memory and communication resources to create a tree in the network. Also fixed base stations become attack targets.
2. Since a far-away base station acts as the intermediary for key management, SPINS is rather energy inefficient. Consider each key request from a sensing node has to traverse multiple hops, reach the base station, and the key traverse all the way back to the requesting node and the destination node.
3. Since the counters and the keys in SPINS are assumed to last for the lifetime of the node, they are vulnerable to being broken either through intercepting the messages (for the keys) or compromising the nodes.

4 Mathematical Analysis

In this section, we perform a 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 C, the hit rate in the cache C_h, and the miss rate C_m. The control cache size is C_c, and its hit and miss rates are C_{ch} and C_{cm}. The control group size is G and the communication group size G_C. The communication group for a node is the neighborhood of that node, with which it *predominantly* communicates. Each node generates one packet every λ seconds on an average, with the inter-arrival time given by an exponential distribution. The destination is chosen at random from the set of communicable nodes. The destination is changed once every μ seconds on an average, again using an exponential distribution. The control node has an average lifetime of τ.. *S(Pkt)* gives the size of a packet. H, H_c, and H_m 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 gives the energy for transmission and reception of one bit. The summary and notations for the different control packets used in SECOS are given in Table 1.

Packet Notation	Description	Packet Notation	Description	Packet Notation	Description
Data	data packet	K_rep	Reply to the key request	control	A broadcast packet 'I am a control node'
change1, change2	Change the master key and the session key respectively	K_repf	Relay reply of the key request, used when one control node sends a key reply to a node in another control group	f_back	feedback packet to the base station
K_req	Request a session key	K_list	A packet holding a list of keys and IDs sent from the base station to the control node	r_flush	Packet from base station to control node ask it to flush its control data and go back to regular sensing mode

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 four factors – security due to control node being compromised, 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.

1. *Security tolerance* (G_{SEC}). We want to limit the amount of communication that will become exposed due to the control node being compromised. Let P be the probability of compromising any node and *N(s)* be the

acceptable number of message communication that can be exposed. The number of possible nodes the adversary can compromise concurrently is *N(C)* = total number of nodes * *P* = *N* * *P*. In the worst case, all these compromised nodes are control nodes. Compromising a control node implies exposing the message communication in sessions that are initiated by nodes in its group in the remainder of the life time of the current control node. We are making the average-case assumption that the control nodes are compromised in the middle of their lifetimes.

$$N(s) \le N.P.G_{SEC}.\frac{\tau/2}{\mu}.C_m \Rightarrow G_{SEC} \le N(s)/(N.P.\frac{\tau/2}{\mu}.C_m) \text{ ———————— (1)}$$

2. *Computational Capabilities (*G_{COMP}). The computational capability of the control node to generate keys for nodes in its group bounds the control group size. Assume that the computational capability of the control node allows it to process *IP* instructions per second and the MAC implementation for Random key generation requires *IK* instructions. The maximum number of keys that can be serviced *is IP*/*IK* keys per second. So if the nodes change destination every μ seconds and the miss rate in the regular cache is C_m, a request is generated once every μ/C_m seconds.

$$G_{COMP} \le \frac{IP}{IK}.\mu/C_m \text{ ———————— (2)}$$

3. *Channel Bandwidth (*G_{BW}*)*. On an average the available bandwidth for each node given channel bandwidth *BW* is *BW/N(NB)*, where *N(NB)* is the number of one-hop neighbors of the node. Given the range of wireless transmission *(r)* and the density of nodes *(ρ): N(NB) = Π* 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 = (*BW/N(NB)*) – (2 * 1/ λ * *S(Data)).*

Each key generation generates *S(K_req)* + 2 * *S(K_rep)* amount of traffic. Taking into account the regular cache misses and the key request rate this term is multiplied by (1/μ . C_m).

$$BW_c \le G_{BW}.((S(K\_req)+2S(K\_rep)).(C_m/\mu)) \Rightarrow G_{BW} \le BW_c/((S(K\_req)+2S(K\_rep)).(C_m/\mu)) \text{ — (3)}$$

4. *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(K)* and the available flash memory is *FM*, then the storage upper bound is given by

$$G_{STORE} \leq FM / S(K) \quad \text{——} (4)$$

The maximum size of the control group is the minimum of those calculated from equations (1), (2), (3), and (4) above.

$$G_{\max} = \min(G_{SEC}, G_{COMP}, G_{BW}, G_{STORE}) \quad \text{——} (5)$$

4.2 Energy-wise Optimal Control Group Size

Here we wish to find the optimal control group size based on security and energy concerns. We want to increase the security by minimizing the number of exposed messages and we want to decrease the overhead energy of the protocol. The security requirement favors decreasing the size of the control group and the smallest size is the best. 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 incidence 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: $H_c = \max(H.\ G/G_C,\ 1)$. From these we derive the following four cases:

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

$$C_h = \frac{C\lambda}{G_c\mu} + \left(1 - \frac{\lambda}{\mu}\right) * \sum_{k=0}^{C} \binom{N-1}{k} \left(\frac{1}{N-1}\right)^k \left(1 - \frac{1}{N-1}\right)^{N-1-k}$$

The first term ($\frac{C\lambda}{G_c\mu}$) represents the probability that the first packet hits and the second term represents the probability that the second hits and so on. We assume that the size of the regular cache is greater than the number of packets sent in μ seconds. However, $C_h = 1$ if the cache size is greater than the communication group size ($C > G_C$). If there is a hit in the regular cache, no overhead energy is spent.

Weighted energy overhead = 0.

Case 2: Miss in the regular cache and the destination is in the same control group. The probability of regular cache miss is $C_m = 1\text{-}C_h$**.** The probability of communication within the same control group is G/G_C.

Weighted energy overhead = Energy overhead per miss. Probability =

$$\left(2 \cdot S(K\_req) + S(K\_rep)\right) \cdot H_c \cdot E \cdot C_m \cdot \left(\frac{G}{G_c}\right)$$

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 $N(G_c) = \lceil G_c / G \rceil$, is given by: $C_{ch} = C_c/(N(G_c)\text{-}1) = C_c/((G_c/G)\text{-}1) = G*C_c/(G_c\text{-}G)$. However, if $G > G_C/(C_c+1)$, $C_{ch} = 1$.

Weighted energy overhead =Energy overhead per miss. Probability =

$$\left\{\left(2 \cdot S(K\_req) + S(K\_rep)\right) \cdot H_c + S(k\_repf) \cdot H\right\} \cdot E.C_m \cdot \left(1 - \frac{G}{G_c}\right)\left(\frac{G \cdot G_c}{G_c - G}\right)$$

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 $C_{cm} = 1 - C_{ch} = 1 - G*C_c/(G_c\text{-}G) = (G_c\text{-}G\text{-}G*G_c)/(G_c\text{-}G)$

Weighted energy overhead = Energy overhead per miss. Probability =

$$\left\{\left(2 \cdot S(K\_req) + S(K\_rep)\right) \cdot H_c + S(k\_repf) \cdot H + \left(2 \cdot S(K\_req) + S(K\_rep)\right) \cdot H_m\right\} \cdot E.C_m \cdot \left(1 - \frac{G \cdot G_c}{G_c - G}\right)\left(1 - \frac{G}{G_c}\right)$$

The total overhead energy of the protocol equals the sum of the contributions of the above four cases. Let size of the key request and reply be the same (R), i.e. $R = S(K\_rep) = S(K\_req) = S(K\_repf)$**.** The total overhead energy T_E is given by:

$$T_E = 3\frac{C_m}{G_c^2}G^2 \cdot R \cdot H \cdot E + C_m \cdot E \cdot \left(1-\frac{G}{G_c}\right)\left(\frac{G \cdot G_c}{G_c - G}\right)\left(\frac{3R \cdot H \cdot G}{G_c} + R \cdot H\right) + C_m \cdot E \cdot \left(1-\frac{G}{G_c}\right)\left(1-\frac{G \cdot G_c}{G_c - G}\right)\left(\frac{3R \cdot H \cdot G}{G_c} + R \cdot H + 3R \cdot H_m\right)$$

$$T_E = C_m \cdot R \cdot H \cdot E\left\{3\left(\frac{G}{G_c}\right)^2 + \left(1-\frac{G}{G_c}\right)\left(\frac{G \cdot G_c}{G_c - G}\right)\left(\frac{3G}{G_c}+1\right) + \left(1-\frac{G}{G_c}\right)\left(1-\frac{G \cdot G_c}{G_c - G}\right)\left(\frac{3G}{G_c}+1+3H_m\right)\right\}$$

By minimizing T_E with respect to G, we get a value of $G = G_{min}$ that minimizes the over head energy of SECOS. This does not give a closed form solution since there are discontinuities due to C_h, C_{ch}, and H_c. We solve the equation numerically, but the solution is not presented here in the interest of space.

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 $G = \min(G_{min}, G_{max})$

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 is picked randomly from the whole network. Four different values of the relative sizes 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 (G)	10
Transmission range	50 m	Ring cache size	20
Number of nodes in the sensor field	200	Regular cache size (C)	0,5,10 entries
The topology in square meters	120X600	Simulation Time	10^5 seconds
Frequency of destination change (μ)	20 seconds	Frequency of master key refreshment	600 seconds
Frequency of packet generation (λ)	5 seconds	Frequency of control node change (τ)	200 seconds
Number of control groups	20	Frequency of session key refreshment	200 seconds
SECOS packet sizes (Bytes): Data (100), change1 (9), change2 (25), Confirm (9), K_req (16), K_rep (16), K_repf (16), K_list (92), R_flush (9), Rd_packet (100)			

Table 2: Simulation Parameters for Evaluation

We measure two parameters for both SECOS and SPINS: the total over head energy due to key management and the average end-to-end delay of data packet. The end-to-end delay of the 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 larger value on the plot implies better performance by SECOS.

In the first experiment, we vary the size of the key 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 SECOS is less favored for the 90%:10% case, due to the occasional choice of destinations far off, outside the control group. Hence, we show the 90%:10% results in Figure 3(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 master and the session keys, and the periodic change of the control node. From these graphs we find that SECOS is always better for both energy and 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. 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 security. 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 till the point where 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 compared to SPINS. However, a reasonable sized control cache as 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 $G_c/G=0.5$ 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.

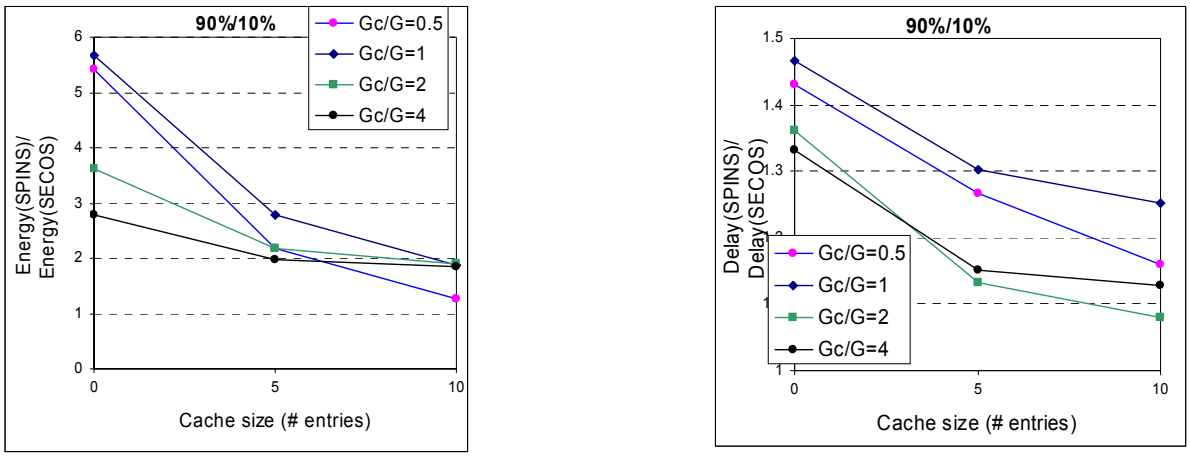


Figure 3: Ratio of (a) overhead energy expended and (b) end-to-end data latency for SPINS and SECOS with varying cache sizes for different communication group sizes

Next, we consider the most general 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 4(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 cache entries, 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 itself. Therefore, with reasonable control cache sizes, SECOS will perform well.

Finally, we bring out the overhead SECOS incurs due to three mechanisms for improving security, namely refreshment of master and session keys, and change of the control node. Figure 4(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.

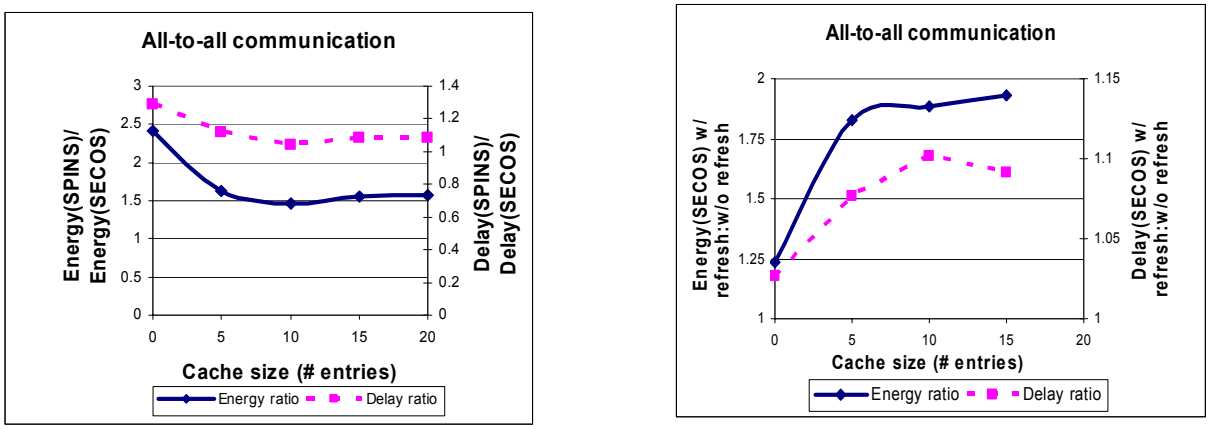


Figure 4: Ratio of overhead energy and delay for (a) SPINS:SECOS (b) SECOS with key refreshment and control node change:SECOS without these techniques

6 Conclusions

We have presented the design of a key management protocol for resource constrained sensor networks, called SECOS. SECOS divides the sensor field into control groups with a control node in each. Data 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. A mathematical analysis is performed to determine the optimal control group size. 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.

In the future, we plan to address 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 pattern which may indicate a security breach. A second problem is determination of the control node. This has to be a trusted entity and it should also have enough resources (such as battery life) 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-election of a control node. This should itself be a protocol which is parsimonious in its energy consumption. We propose to use collaborative monitoring of a sensor node's behavior by its neighbors to determine the trustworthiness of the node.

References

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

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

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

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

[10] 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.

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

[12] 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.

[13] D. Liu, 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.

[14] S. Basagni, K. Herrin, D. Bruschi, 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.

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

Appendix

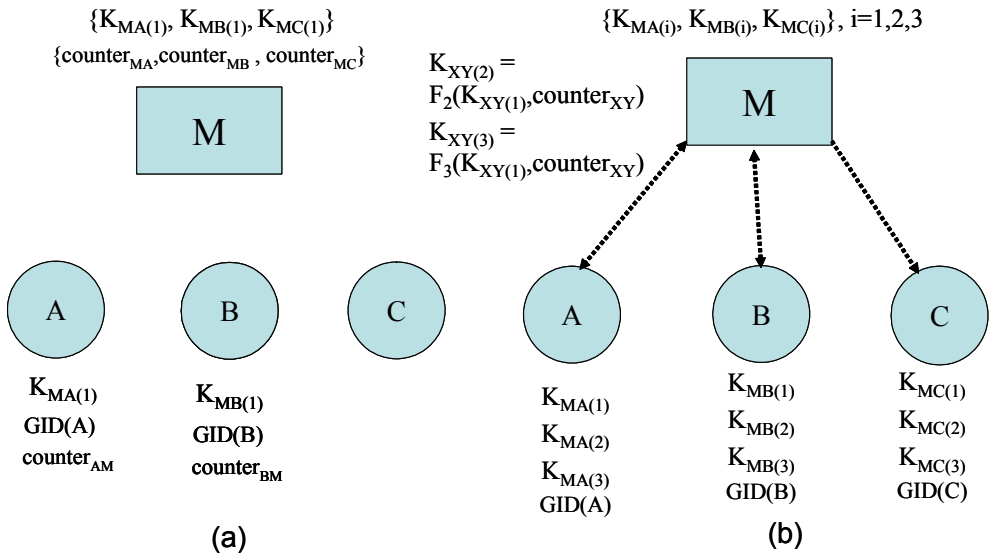


Figure 5: Initial key setup between base station and three sensing nodes

Node-Node Key establishment within the control group:

A node A wants to communicate with a node B, and both A and B share the same control group C.

$A \rightarrow C$: $A,B,MAC(K_{AC(3)}, A|B|Counter_{AC})$

$C \rightarrow A$: $K_{AC(2)}(K_{AB(2)}),MAC(K_{AC(3)}, K_{AC(2)}(K_{AB(2)})| Counter_{CA})$

$C \rightarrow B$: $K_{BC(2)}(K_{AB(2)}),MAC(K_{BC(3)}, K_{BC(2)}(K_{AB(2)})| Counter_{CB})$

Node-to-Node key establishment between control groups :

A node A in control group C_1 wants to communicate with a node B in control group C_2:

1- A sends a key request Packet to its control node C_1

$A \rightarrow C_1$: $A,B,MAC(K_{AC1(3)}, A|B|Counter_{AC1})$

2- C_1 checks its control cache for C_2, if an entry exists go to step 4. Else send a key request to the base station for a common key with C_2

$C_1 \rightarrow M$: $C_1,B,MAC(K_{MC1(3)}, C_1 |B|counter_{MC1})$

3- The base station replies with a session key sent to both C_1 and C_2

$M \rightarrow C_1$: $K_{MC1(2)}(K_{C1C2(2)})$,MAC($K_{MC1(3)}$, $K_{MC1(2)}(K_{C1C2(2)})$| $Counter_{MC1}$)

$M \rightarrow C_2$: $K_{MC2(2)}(K_{C1C2(2)})$,MAC($K_{MC2(3)}$, $K_{MC2(2)}(K_{C1C2(2)})$| $Counter_{MC2}$)

4- C_1 generates a common session key for A and B and send one copy to C_2 to be forwarded to B and another copy to A

$C_1 \rightarrow C_2$: A,B, $K_{C1C2(2)}(K_{AB(2)})$,MAC($K_{C1C2(3)}$, A|B| $K_{C1C2(2)}(K_{AB(2)})$|$counter_{C1C2}$)

$C_1 \rightarrow$ A : $K_{AC1(2)}(K_{AB(2)})$,MAC($K_{AC1(3)}$, $K_{AC1(2)}(K_{AB(2)})$| $Counter_{AC1}$)

5- C_2 forward the key to B

$C_2 \rightarrow$ B : $K_{BC2}(K_{AB(2)})$,MAC($K_{BC2(3)}$, $K_{BC2(2)}(K_{AB(2)})$| $Counter_{BC2}$)

Master Key refreshment:

The master key between the base station M and a sensor S is to be refreshed:

1- M sends to S a packet called $change_1$ ordering it to generate the next master key

$M \rightarrow S$: $change_1$,MAC($K_{MS(3)}$,$change_1$|$counter_{MS}$)

2- In response to the order, S generates the next master key and sends a done packet to M

S : new($K_{MS(1)}$) = F_1($K_{MS(1)}$, $K_{MS(1)}$)

$S \rightarrow M$: done,MAC($K_{MS(3)}$,done|$counter_{SM}$)

3- When M receives the done packet, it generates the corresponding new master key and all other keys are generated from it

M : new($K_{MS(1)}$) = F_1($K_{MS(1)}$), reset counters

new($K_{MS(2)}$) = F_2($K_{MS(1)}$,$counter_{MS}$)

new($K_{MS(3)}$) = F_3($K_{MS(1)}$, $counter_{MS}$)

4- M sends a final confirmation to S.

$S \rightarrow M$: confirm, MAC($K_{MS(3)}$,confirm|$counter_{MS}$)

5- When S receives the confirmation, it resets the counters and generate the relevant keys from the new master key

S: delete the old $K_{MS(1)}$, reset the counters

$new(K_{MS(2)}) = F_2(K_{MS(1)},counter_{SM})$

$new(K_{MS(3)}) = F_3(K_{MS(1)}, counter_{SM})$

Session and Authentication Key refreshment:

The encryption and authentication keys between M and S are to be refreshed:

1- M generates a random value (value)

$value = F_4(K_{MS(4)},counter_{MS})$

2- M sends an order called $change_2$ to S to refresh the session key. This order also caries the control node C for S and the random value (value)

$M \rightarrow S : change_2,C,value,MAC(K_{MS(3)}, change_2 |C|value)$

3- When S receives the order it refreshes the session and the mac keys

S : $new(K_{MS(2)}) = F_2(K_{MS(1)}, value)$

$new(K_{MS(3)}) = F_3(K_{MS(1)}, value)$

4- S sends a done message to M

$S \rightarrow M : change2,done,S,MAC(K_{SM(2)},change2|done|S|counter_{SM})$

5- When M receives the done message it refreshes the session and the MAC keys with S

M : $new(K_{MS(2)}) = F_2(K_{MS(1)}, value)$

$new(K_{MS(3)}) = F_3(K_{MS(1)}, value)$

This process can be done between any pair of nodes having secure sessions.

Control Node refreshment:

The base station M refreshes the control node C of a certain control group G:

1- M selects a new (may be the same) control node

2- M generates a new session key of each node in that control group using the last shared random value with each node

$new(K_{iC(2)}) = F_2(K_{iM(1)}, value+1)$, save the new value

3- M sends to C a list of these generated session values

$M \rightarrow C : K_{MC(2)}(\{K_{iC(2)}, ID_i\}), MAC(K_{MC(3)}, K_{MC(2)}(\{K_{iC(2)}, ID_i\})|cntr_{MC})$

4- The new control node announces its presence

$C \rightarrow G$: I am a control node for group G

5- S save C, while keeping the old one until C is authenticated

6- S (when need C): Challenge C

Sensor to Control Challenge:

A node S challenges a new control node C. (value is the last shared random value):

1- S generates the session key ($K_{SC(2)}$)

$K_{SC(2)} = F_2(K_{MS(1)}, value+1)$

$K_{SC(3)} = F_3(K_{MS(1)}, value+1)$

2- S generates a random number (RN)

$RN = F_4(K_{SM(4)}, counter)$

3- S sends the random number encrypted with the session shared session key that the new control node must have if it is really the new control node.

$S \rightarrow C : K_{SC(2)}(RN), MAC(K_{SC(3)}, K_{SC(2)}(RN))$

4- The control node decrypt the message increment the random number, encrypt the new number and send it back to the challenger

$C \rightarrow S : K_{SM(2)}(RN+1), MAC(K_{SC(3)}, K_{SC(2)}(RN+1))$