Multi-level Security in Wireless Sensor Networks

Faruk Bagci and Theo Ungerer
Institute of Computer Science
University of Augsburg
Augsburg, Germany
{Bagci, Ungerer}@Informatik.Uni-Augsburg.DE

Nader Bagherzadeh
Department of Electrical Engineering and Computer Science
University of California
Irvine, USA
nader@uci.edu

***Abstract*—As the potential range of applications for sensor networks expands, the need for security mechanisms grows. Security relevant problems are limited mostly to areas such as key distribution and cryptographic algorithms, due to severe resource constraints in wireless sensor networks. Even if it is not possible to cover all threats, a security architecture for sensor networks should provide mechanisms on several levels, in order to maximize the obstacles for attackers. This paper presents *SecSens*, an architecture that provides basic security components for wireless sensor networks on multiple system levels. Since robust and strong security features require powerful nodes, SecSens uses a heterogeneous sensor network. In addition to a large number of simple (cheap) sensor nodes providing the actual sensor tasks, there are a few powerful nodes (cluster nodes) that implement the required security features. The basic component of SecSens offers authenticated broadcasts to allow recipients to authenticate the sender of a message. On the basis of this basic component, SecSens provides a key management, used for exchange of secret keys among nodes. In order to bind nodes to their neighborhood, keys and their owners are linked together. This ensures a certain grade of location relationship. To protect the Sensor network against routing attacks, SecSens includes a probabilistic multi-path routing protocol, which supports the key management and the authenticated broadcasts. SecSens also provides functions to detect forged sensor data by verifying data reports en-route. In order to evaluate the efficiency of our security architecture, we simulated different sizes of sensor networks. Furthermore, SecSens is successfully evaluated in a real test environment with two different kinds of sensor boards.**

***Keywords*-wireless sensor network; security architecture; key management; energy efficiency; multi-path routing; en-route filtering**

I. Introduction

Wireless Sensor Networks (WSN) have emerged as a new information-gathering paradigm based on the collaborative efforts of a large number of self-organized sensing nodes. These networks form the basis for many types of smart environments such as smart hospitals, intelligent battlefields, earthquake response systems, and learning environments. A set of applications, such as biomedicine, hazardous environment exploration, environmental monitoring, military tracking and reconnaissance surveillance, are the key motivations for the recent research efforts in this area [7] [9].

Different from traditional networks, sensor networks do impose a set of new limitations for the protocols designed for this type of networks [10]. Devices in sensor networks have a much smaller memory, constrained energy supply, less process and communication bandwidth. Topologies of the sensor networks are constantly changing due to a high node failure rate, occasional shutdown and abrupt communication interferences. Because of the nature of the applications supported, sensor networks need to be densely deployed and have anywhere from hundred to thousands of sensing devices, which are orders of magnitude larger than traditional ad hoc mobile networks. In addition, energy conservation becomes the center of focus because of the limited battery capacity and the difficulty of recharge in the hostile environment. With fundamental difference between traditional networks and sensor networks, it is not appropriate and probably inefficient to port previous solutions for ad hoc networks into sensor networks with only incremental modifications. For instance, the sheer number of sensor nodes makes flooding-based standard routing schemes for ad hoc networks undesirable [4].

Because of the steady increase in applications, security requirements for sensor networks have received more attention. Areas such as health or safety critical industrial facilities offer very good use for sensor networks, on the other hand, they also demand high safety standards to be observed. A security architecture can never cover all types of threats simultaneously. The application determines, which attack vectors are probable in current scenarios, and how attractive collected and processed data could be for a potential intruder. In particular, denial-of-service attacks at network level, require special and expensive countermeasures. A comprehensive security architecture can increase protection and number of blocked attacks, but on the other hand, hardware costs and thus cost per sensor board increases, which is not always desirable. It also may increase energy requirements of sensor nodes significantly due to several successive protocols. This extra effort can, however, reduce considerably the life-time of individual sensor boards.

Not every sensor network pays an attack with enormous resources required to access its data, or to block it. In most cases, a combination of multiple protocols can confine a wide range of threats. Therefore a good compromise between cost and protection is often not a full defense against all attacks, instead it is preferable to maximize the obstacles for attackers. Many attackers resile if they have to increase extremely costs to break a security architecture.

Regarding sensor networks, following basic requirements exist for the security concept:

- **Confidentiality:** Confidentiality means that information remain secret to unauthorized parties. Therefore sensor nodes must protect transferred data against illegal access.
- **Authenticity:** A node must always verify the authenticity of received messages. In particular, authenticity ensures that messages are really sent by the stated source.
- **Integrity:** Received data can be changed during transmission by failures or intent. Integrity should recognize these manipulations.
- **Timeliness:** The timeliness of data ensures that received information is up-to-date. An attacker should not be able to send old data (repeatedly).
- **Scalability:** Especially key management in large sensor networks can cause significant burden. Therefore the security architecture should consider also scalability.
- **Availability:** The sensor network should be robust and fault-tolerant, *i.e.* compromising individual node should not affect security of the entire network. On the other hand, the effort for security must not impair actual tasks of the sensor network, *e.g.*, by long delays.
- **Compromise:** A complete protection can not be ensured by any security architecture. Hence it must take the worst-case scenario into account, in which parts of the network are compromised. The aim should be a verification of data in order to prevent insider attacks, or at least to limit them locally.
- **Self-organization:** Characteristic for sensor networks is their capacity for self-organization. This should be considered by the security concept, especially in relation to key management and join/loss of nodes in the network.
- **Accessibility:** In order to decrease transfer costs, intermediate nodes on a path to the base station should aggregate and process received sensor data (*in-network-processing/ aggregation*). This implies, however, suitable access to secured data. By contrast, access must be minimized as far as possible to limit possible impact of compromised nodes.
- **Energy-efficiency:** To provide a certain life-time and to prevent attacks on depletion of energy resources, energy-efficiency plays an important role in security architecture that should be taken into account.

This paper describes *SecSens* a security architecture for wireless sensor networks that fulfills above mentioned requirements [1]. SecSens focuses mainly on a robust and secure routing protocol and protection against data manipulation. The next section describes related security approaches for sensor networks. Section III introduces the SecSens architecture and its security features. SecSens is successfully evaluated in a simulator and a real test environment with two different kinds of sensor boards. Section IV describes the evaluation results. The paper ends with the conclusion.

II. Related Work

SecSens focuses basically on three security aspects: key management, secure routing, and verification of sensor data. This section describes related approaches that are relevant for each aspect and can be used in wireless sensor networks.

Critical factor in key management is secure and efficient distribution of keys to sensor nodes. Because of limited resources in sensor networks usually symmetric keys are used. Symmetric encryption requires that both communicating nodes know the same secret key.

Key Management: [2] presents secure key distribution techniques for sensor networks. In particular, two approaches are described: single network-wide key and pair-wise shared key. The simplest method of key distribution is to pre-load a single network-wide key onto all nodes before deployment. Storage cost is minimal because each node has to store only one key. Unfortunately this approach provides sufficient security only if all nodes are protected against physical force. But this usually does not apply to low cost sensor nodes. The pair-wise shared key approach requires that every node in the sensor network shares a unique symmetric key with every other node. Hence, in a network of n nodes there are a total of $\binom{n}{2}$ unique keys, whereas each node has to store $n-1$ keys. The storage cost is proportional to the total number of nodes in the network. Therefore, the pairwise key scheme does not scale well for large sensor networks.

In [8] a security protocol for sensor networks called *SPINS* was presented for hierarchical sensor networks with one or more trustworthy base stations. SPINS consists of two parts: a secure network encryption protocol (SNEP) and authenticated broadcasts (μTESLA). Each sensor node receives on a secure channel an individual, symmetric master key, which is only known by the base station and the node. Using this master key the sensor node is able to generate all keys. The disadvantage of SNEP is that secure communication can be built only between a base station and nodes, and it is not possible to protect the communication in or between clusters. The second part of SPINS is μTESLA that provides sending of authenticated broadcasts. For symmetric encryption, sender and receiver must share the same secret. Consequently, a compromised receiver is able to act as a designated sender by transferring forged messages to all receivers. μTESLA uses delayed disclosure of symmetric keys for generating an asymmetry between sender and receiver. This approach requires weak time synchronization of sender and receiver in order to achieve time shifted key disclosure. Storage cost increases because each node has to buffer packets which it can only verify after receiving the key in the future time-slots. Also, this causes new possibilities for DoS-attacks. An attacker can force a buffer overflow by sending planned broadcasts. Furthermore μTESLA leads to scalability problems, which are described in [6].

An important aspect for sensor networks is that different communication patterns exist requiring different security steps. [13] suggests an adjusted key distribution for different security requirements. For this reason, four different kinds of keys are used. The individual key is similar to the master key of SPINS. The second kind of key is a pair-wise shared key, which is generated in the initial phase for each known neighbor. Furthermore, the nodes have a cluster key for secure communication between cluster members. The last key is the group key that is used for secure broadcasting. This approach provides more flexibility but contains a security risk during the initial key distribution phase.

Secure Routing: Compromised sensor nodes can influence a sensor network, especially by manipulating of routing information. In order to minimize the impact, [3] suggests intrusion tolerant routing in wireless sensor networks called *INSENS*. The goal is to provide a working network even if parts of it are infiltrated. INSENS contains two phases: route discovery and data forwarding. In the first phase, the base station sends out a broadcast to build routes to each node. After receiving the route request, the nodes send a list of all known neighbors back to the base station. For the last step, the base station generates several disjoint paths to each node and sends this routing information back to all network members. Based on this routing table, the nodes can forward data to the base station (data forwarding phase). INSENS prevents the network against most outsider attacks and even insider attacks remain locally. But the dependance on the base station suggests a single point of failure. Furthermore, the route discovery phase is extremely energy inefficient.

Another approach for secure routing called *ARRIVE* is presented in [5]. The routing algorithm tries to send packets over different paths based on probability. Also, nodes in ARRIVE listen passively to communication among neighbors. In case of detected failures, other nodes can forward the packet on behalf of its neighbor. ARRIVE works with smaller routing tables, but the chosen path is not always optimal. Furthermore, ARRIVE does not provide authenticated broadcasts, that provides a mechanism for manipulation of routing information.

Verification of Sensor Data: Each individual sensor node is potential target for attackers. Using compromised nodes, an attacker can directly influence the sensor network by infiltrating false reports of network sensor data. This kind of attack is called fabrication report attack. These fake reports can reach the base station, if they remain undetected, where they can trigger off false alarms. Also this causes high consumption of bandwidth and energy. En-route filtering attempts to verify reports on the way from sending node to the base station. The goal is to detect and discard false reports earlier. [14] describes an interleaved hop-by-hop authentication scheme for filtering of injected false data. This algorithm recognizes infiltrated reports by a deterministic process, as long as no more than t nodes are compromised. The sensors build clusters with a minimum of $t+1$ nodes, where each group chooses a cluster head C. Only the cluster heads can send collected events in the form of reports to the base station. These reports include additionally to the actual event $t + 1$ independent confirmations of the respective parties, in order to verify the authenticity of the report. Each intermediate node checks the report on the way to the base station (*En-route filtering*). Unfortunately, this approach uses single path routing to the base station providing several security risks. A statistical en-route filtering is presented in [12] that enhances the approach above by using probabilistic algorithms. Each node chooses randomly a number of keys from a partition of a global key pool. Because of the probabilistic distribution of keys, any node can verify with a certain probability a report before it is forwarded. In this manner, [12] supports multi-path routing. But in both approaches, an attacker can create any report, once it has compromised at least t nodes. [11] attempts to solve this problem by binding keys to the location of nodes. The sensor area is divided into cells of width c, whereas each cell contains several keys. The nodes receive a location-bound key for each *sensing cell*. This approach bounds reports to their original location.

III. SecSens - Security Architecture for Wireless Sensor Networks

The sensor network in SecSens consists of clusters, each containing simple sensor nodes u_i and one powerful sensor node v that acts as a cluster-head. Sensor nodes u_i connect directly to the cluster-head, because routing in clusters is not necessary. Sensor nodes can be a member of several clusters. Cluster-heads again build together an inter-cluster network, that is used to transfer messages to base stations. It is assumed that sensor nodes have a fixed position, once they are attached to a location. SecSens works with multiple base stations to avoid the risk of single-point-of-failure (see Figure 1).

The security architecture of SecSens combines several security approaches in order to provide high protection. Basically SecSens contains four components, which interact with each other: authenticated broadcasts, key management, routing, and en-route filtering.

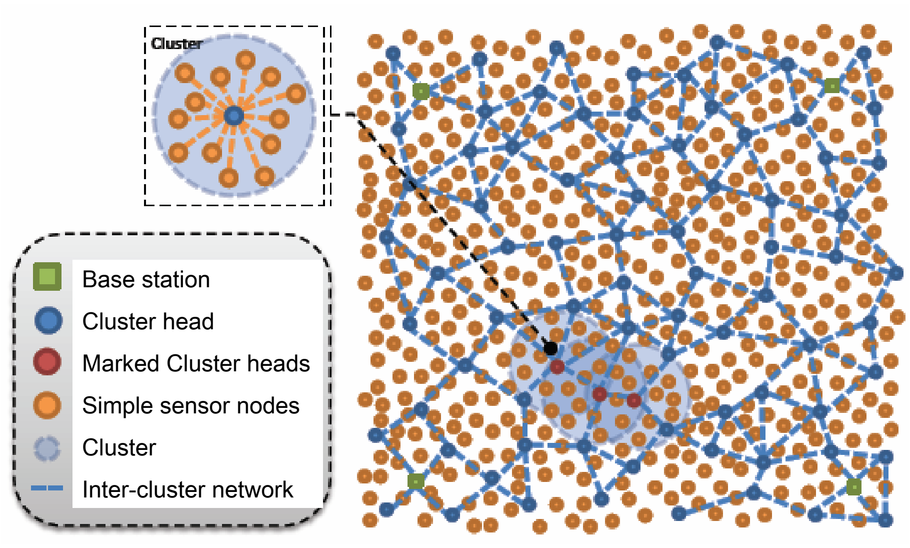


Figure 1. Basic sensor network architecture

A. Authenticated Broadcasts

Authenticated broadcasts ensure that the stated sender is identified as the true sender. Symmetric approaches use a shared key to generate a *message authentication code (MAC)*. In case of only one receiver, the sender is clearly identified. But if there are multiple receivers with the same shared key, this approach for authentication is not applicable. Potentially each receiver could be the sender. To solve this problem, there must exist an asymmetry between sender and receiver.

SecSens provides two authenticated broadcasts: broadcasts from base stations, and broadcasts in clusters. It is assumed that base stations are trustworthy and can not be infiltrated. In order to generate an asymmetry, SecSens uses key chains. Each packet contains a key. To decrypt a previous packet, a node has to wait for the key of next packet (Figure 2).

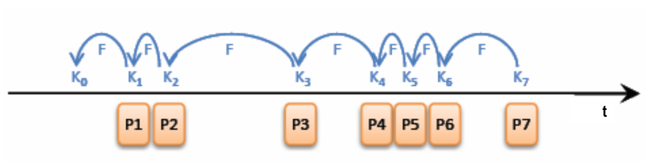


Figure 2. Key chain approach

The base station generates a key chain $K_0^b \dots K_n^b$ with sufficient number of keys using a publicly known one-way function F, so that for $i \in \{0, \dots, n-1\}$ is:

$$K_i^b = F(K_{i+1}^b) \tag{1}$$

Each node knows that the first key in the chain is $\langle i, K_i^b \rangle$ with $i = 0$. Therefore, K_0^b is public and K_1^b is the first non-disclosed key. Keys can be used only once. In order to broadcast a message M, the base station calculates the corresponding MAC using the next non-disclosed key K_i^b and sends it together with used key index to all its neighbors:

$$BS \rightarrow * : \qquad i, MAC_{K_i^b}(M) \tag{2}$$

A receiving node u checks, if it has already received a MAC for the stated index, before storing the MAC and index i into the MAC buffer. Consequently node u accepts only one MAC per key index. If i is a new index and u is a cluster-head, it forwards the message to all its neighbors. In this way, it is efficiently distributed over the inter-cluster network to all sensor nodes. After a maximum time T all sensor nodes know the MAC together with its key index. Sensor nodes can not manipulate the MAC, because the base station has not yet disclosed K_i^b at this time. Time T_p describes a dynamically adaptable system parameter. The base station can set T_p depending on the network size, whereas $T < T_p$. After expiration of T_p, the base station sends the actual message M besides the disclosed key data $\langle i, K_i^b \rangle$ to all neighbors:

$$BS \rightarrow * : \qquad M, \langle i, K_i^b \rangle \tag{3}$$

Sensor node u can now verify K_i^b using a previous disclosed key K_{i-1}^b. If it does not have K_{i-1}^b, it can verify the current key through recursively performing Equation 1 with previous keys. To prevent DoS-attacks, the new key must not be older than G_{max} generations. If K_i^b is finally verified, node u makes new key data $\langle i, K_i^b \rangle$ effective. Subsequently, u can check message M using $MAC_{K_i^b}(M)$ and index i. It is important that the index for MAC and key are the same.

The concept of authenticated broadcasts for base stations has a disadvantage, if it is used for local authenticated broadcasts between clusters, because of the delayed disclosure of keys. However, all receivers of a message in a cluster can be reached after a single hop. This fact can be used to simplify the concept, in order to avoid time delays. The cluster-head generates a key chain $K_0^b \dots K_n^b$ using a publicly known function F (see Figure 2). All cluster members receive the first key K_0^b over a secure connection using pair-wise shared keys known only by cluster-head and respective node. Cluster-head v uses for each local authenticated broadcast a key K_i^b, which is not yet disclosed. It sends the message M together with the key data:

$$v \to * : \qquad M, K_i^b \tag{4}$$

Cluster nodes can verify K_i^b and the message M using previously disclosed keys. With this the sender is authenticated as cluster-head, because only the cluster-head can know K_i^b, that was not public until current message. On the other hand, no intermediate node can manipulate the message M, because all potential receivers of M are reachable with only one hop.

B. Key Management

Sensor nodes basically distrust each other. To build trust between two or more nodes, a shared secret in the form of keys is needed. However, neighborhood and relations between sensor nodes are not known before. Therefore, nodes must build trust during life-time, more strictly in the initial phase. For security reasons, a sensor node should join a network only once. In this critical phase, it can establish shared keys with its neighbors. Since this procedure is performed only once, the node is binding itself to the location. Furthermore, sensor nodes can communicate at several levels, that is cluster- or network-wide. Consequently, SecSens uses several kinds of keys to fulfill different security requirements.

Activation: In the initial phase, a sensor node needs an initial key K_I. This key is stored only on a specific activation node, which does not take part for usual networking tasks. The basic idea is that sensor nodes can not install themselves independently. Instead a trustworthy employee, who own the activation node, establishes sensors. In order to add a set of sensor nodes, the base station stores a randomly generated master key K_A, timer T_A, initial key K_I, and current group key K^g onto the activation node A. Using master key K_A, the base station can generate for each sensor node u a personal activation key K_u^a based on the node ID.

$$K_u^a = F_{K_A}(u) \tag{5}$$

All key material and other security critical data is stored only in RAM of A. T_A determines period of validity for the master key. After expiration of this time, the activation node deletes K_A and all critical data. Each sensor node u has a unique ID and a personal activation key K_u^a. In order to activate sensor node u, the activation node A has to be in the communication range. For security reasons, the radio power of A is kept low, to ensure physical proximity. After turning on for the first time, sensor node u broadcasts periodically its plain ID and the ID encrypted with the personal key with the same low radio power. Activation node A can easily verify the ID, because it knows the master key K_A. As the next step, A encrypts with personal key of u the initial key K_I, group key K^g, and data X that was given by base station. Finally, A sends encrypted message to sensor node u:

$$u \to A : \qquad u, MAC_{K_u^a}(u) \tag{6}$$
$$A \to u : \qquad \{K_I, K^g, X\}_{K_u^a} \tag{7}$$

After receiving all key material, sensor node u is activated and it deletes the personal activation key.

Group keys: The group key K^g is used by the base station to secure network-wide communication. An attacker, who compromises a node, can also access the group key. In order to update K^g, the base station broadcasts first a list of known compromised nodes $\{x_1, ..., x_m\}$ to all sensor nodes. Additionally, it sends a verification key $F_{K^{g\prime}}(0)$, whereas $K^{g\prime}$ is a new randomly generated group key, and F a publicly known one-way function. $F_{K^{g\prime}}(0)$ is used later to verify the new group key $K^{g\prime}$.

$$BS \to * : \qquad i, MAC_{K_i^b}(\{x_1, \dots, x_m\} || F_{K^{g\prime}}(0)) \tag{8}$$
$$BS \to * : \qquad \{x_1, \dots, x_m\}, F_{K^{g\prime}}(0), \langle i, K_i^b \rangle \tag{9}$$

The base station uses an authenticated broadcast with key K_i^b and index i that is not disclosed yet. After receiving key K_i^b and successfully verifying the above message (see section III-A), sensor nodes delete all pair-wise shared keys or cluster-keys with compromised node x_i. Cluster-heads additionally update their cluster-keys and inform other non-compromised cluster-heads about new cluster-key. Afterwards, all sensor nodes store verification key $F_{K^{g\prime}}(0)$. As a second step, the base station publishes new group key $K^{g\prime}$. Therefore, it encrypts the group key using its cluster-key K_{BS}^c and transfers the message to all direct neighbors. The neighbors can verify $K^{g\prime}$ using verification key $F_{K^{g\prime}}(0)$ and store afterwards the new group key.

$$BS \to * : \qquad \{K^{g\prime}\}_{K_{BS}^c} \tag{10}$$

If receiver u is a cluster-head, it forwards new group key $K^{g\prime}$ encrypted by its own cluster key K^c_u. Consequently, the new group key $K^{g\prime}$ is forwarded over the inter-cluster network to all sensor nodes. Since cluster-heads updated their cluster keys before, the compromised nodes do not receive the new group key. This procedure is periodically repeated by base stations to prevent the network against attacks. If there are no new known compromised nodes, the transferred list is empty.

Pair-wise shared keys: For secure communication between sensor nodes, pair-wise shared keys are used. A new cluster-head exchanges a pair-wise key with all neighbors. Simple sensor nodes communicate only over cluster-head, therefore, they need only a shared key with their cluster-head. Using the initial key K_I each node u generates a personal master key K^p_u based on its ID. In order to establish a pair-wise shared key with its neighbor v, node u needs the ID of v. For this reason u broadcasts a HELLO-message containing its ID. If v decides to establish secure connection with new node u, it answers with an acknowledgment containing its own ID.

$$u \rightarrow * : \quad u \tag{11}$$
$$v \rightarrow u : \quad v, MAC_{K^p_v}(u||v) \tag{12}$$

The additional MAC authenticates the acknowledgement of v, because u can calculate master key K^p_v of v using initial key K_I. Node u does not need to authenticate itself, because the succeeding message exchange verifies the identity of u. The pair-wise shared key K^p_{uv} can be calculated by both nodes without new message exchange:

$$K^p_{uv} = F_{K^p_v}(u) \tag{13}$$

After expiration of time T_I the nodes delete initial key K_I and all personal master keys of its neighbors received during initialization. Only own master key is stored for future pair-wise keys with new sensor nodes. The annulment of compromised pair-wise keys is efficiently realized by deletion of corresponding keys.

Cluster keys: Sensor nodes transfer information to all other cluster members using the cluster key without encrypting the message for each receiver separately. This approach allows in-network-processing and passive participation of sensor nodes within a cluster. Cluster-head u generates randomly cluster key K^c_u, if u joins a network or if u updates cluster key because of compromised nodes. Each cluster member $v_1, \ldots, v_m$ receives new cluster key K^c_u, whereas u encrypts cluster key using pair-wise shared keys $K^p_{uv_i}$ for $i \in \{1, \ldots, m\}$:

$$u \rightarrow v_i : \quad \{K^c_u\}_{K^p_{uv_i}} \tag{14}$$

Only sensor node v_i can decrypt cluster key and store it. If an additional sensor nodes v joins the network, it establishes a new pair-wise shared key with cluster-head. In this case it also gets the current cluster key. If a cluster member is compromised, cluster-head annuls cluster key K^c_u and distributes new key $K^{c\prime}_u$ as described above, without sending it to the compromised nodes.

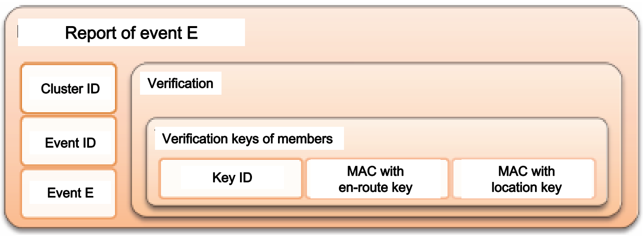


Figure 3. Report content

C. Routing

To prevent attacks on routing level or restrict them locally, SecSens provides a secure routing protocol. Simple sensors in SecSens do not need routing capability, because they exclusively communicate with the cluster-head. Therefore, routing is used only within the inter-cluster network built by cluster-heads. The routing algorithm has two phases: initialization and the actual routing. In the initialization phase each node gets a level using breadth first search that determines the distance to base station in hops. Since base station has level 0, its direct neighbors have level 1. The base station uses an authenticated broadcast including its ID BS_u, a non-disclosed key K^b_i, and key index i to authenticate an initialization.

$$BS \rightarrow * : \quad i, MAC_{K^b_i}(BS_u) \tag{15}$$
$$BS \rightarrow * : \quad BS_u, \langle i, K^b_i \rangle \tag{16}$$

Cluster-heads can identify from authenticated messages, which base station wants to update routing information. After reception of initialization, cluster-heads have time T to modify their routing tables. After expiration of T, further changes are not allowed. Cluster-heads set their level on $L = \infty$ after reception of the message (see Equation 16). The breadth first search can now begin. Starting from the base station, the level values are locally broadcasted by cluster-heads. To prevent outsider-attacks each cluster-head u uses key K^b_i, that will be published later, and its cluster key K^c_u to generate an encrypted message containing ID and level value L_u:

$$u \rightarrow * : \quad \{u||L_u\}_{K^c_u}, K^b_i \tag{17}$$

A cluster-head v updates its level to $L_v = L_u + 1$, if $L_v > L_u + 1$. It also stores level of u. After level update, v forwards its level value to its neighbors in the same way. Each base station triggers its own initialization without disturbing ongoing updates of others. The cluster-heads manage a routing table for each base station.

SecSens uses probabilistic multi-path routing based on the level values to forward messages from cluster-heads on the way to the corresponding base station. Cluster-heads build up a trust matrix, where each transmission to its neighbors is recorded. Based on this trust information and current level, cluster-heads calculate a probability value and write it into the packet header. This value is used to decide in which direction the packet has to be send. Each cluster-head modifies the probability value and sends the message over the most trustworthy route. Since this could lead to the problem that a packet stays at the same level while making a round-trip, the weight of upper level increases with each hop. This ensures that packet transfer goes in the direction of the base station.

Furthermore, SecSens provides passive participation, *i.e.* sensor nodes listen to packet transmissions of their neighbors. If cluster-head u detects a packet addressed to its neighbor v, and recognizes that v is not forwarding the message, u takes responsibility with a certain (low) probability. Also, if u assumes that v forwards the message to a non-existent node, u takes care of transferring.

D. En-route Filtering

Attacks like *report fabrication* or *false data injection* threaten the network by manipulating and infiltrating sensor data. SecSens prevents such threats using en-route filtering extending approach of [12]. Cluster-heads generate data reports containing sensor information of cluster members for sending them to base stations. These reports are verified during transfer through the inter-cluster network (see Figure 3).

En-route filtering consists of three phases: key generation, report generation, and verification.

Key generation: SecSens provides a global pool containing N en-route keys $\{K_0^e, \ldots, K_{N-1}^e\}$. The keys K_i^e are subdivided in n partitions with each m keys. Each sensor node generates all en-route keys in the initial phase using one-way function F and chooses randomly a partition j where it finally draws $k < m$ keys from set j:

$$\forall i \in \{0, \ldots, N-1\} : K_i^e = F_{K_M^e}(i) \qquad (18)$$

At least base stations can detect all fault reports, because they have global view of the key pool. Based on the same partition j, each node calculates in a similar way location key $K_{C,j}^l$ for all its clusters that it senses as a member. Sensor nodes bind themselves locally to actual cluster by the location key. After the initial phase nodes delete all remaining unused keys.

Report generation: If a cluster-head wants to generate a report, it collects sensor data from all its cluster members. Sensor nodes belonging to same cluster report events (sensor data) collectively by generating MACs based on their en-route keys, whereas keys must be chosen from different partitions. These multiple MACs collectively act as the proof that a report is legitimate. Finally, the cluster-head forwards the report to the base station over the inter-cluster network.

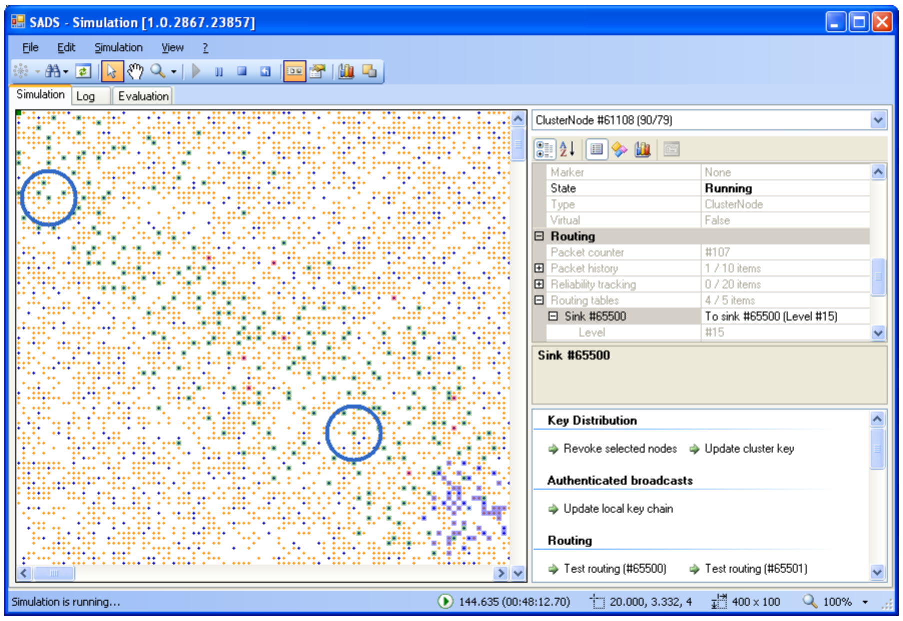
Verification of reports: A cluster-head receiving a report checks, if it has one of the keys, that were used to generate the MACs in the report. With a certain probability, it will be able to verify the correctness of MACs. A report with an insufficient number of MACs will not be forwarded. A compromised node has keys from one partition and can generate MACs of one category only. Since keys and indices of distinct partitions must be present in a legitimate report, the compromised node has to forge the other key indices and corresponding MACs. This can be easily detected by cluster-heads possessing these keys. If cluster-head has none of the keys and number of MACs is correct, it forwards report to the next cluster-head. Even if a forged report receives a base station, it can be detected, because base stations know all used keys.

IV. Evaluation

To evaluate the efficiency of our security architecture we implemented a simulation tool where it is possible to establish different sizes of sensor networks. Figure 4 shows the GUI of the SecSens simulator. The aim was to measure energy consumption and throughput of security mechanisms. SecSens simulator provides the possibility to change parameters like network size, node density, and number of cluster heads or cluster members. After initializing the network we performed several attacks and examined the network stability.

Denial-of-Service attacks are the most common threats in a network. SecSens limits the effect of DoS attacks to a local area compensating high node failures. When a node failure is detected, SecSens tries to send the message over an alternative route. Figure 5 shows the delivery rate of messages in dependance to node failure varying the distance between failure location and receiver. We simulated here a sensor network with 10.000 nodes. If the center of failure is near to message receiver (5 hops), SecSens finds enough alternative routes to achieve %50 delivery rate considering number of 100 failed nodes. Increasing the distance means that the failure center comes near to the message sender blocking most alternative routes.

Insider attacks aim to manipulate behavior of the sensor network by infiltrating false sensor data. SecSens uses data reports to secure the delivery. In order to generate a new report, a node needs five verification keys from five different neighboring nodes. If an attacker wants to send a false report, he has to compromise several nodes to access the



ClusterNode #61108 (90/79)

Marker None
State Running
Type ClusterNode
Virtual False
Routing
Packet counter #107
Packet history 1 / 10 items
Reliability tracking 0 / 20 items
Routing tables 4 / 5 items
Sink #65500 To sink #65500 (Level #15)
Level #15

Sink #65500

Key Distribution
Revoke selected nodes
Update cluster key
Authenticated broadcasts
Update local key chain
Routing
Test routing (#65500)
Test routing (#65501)

Simulation is running... 144.635 (00:48:12.70) 20.000, 3.332, 4 400 x 100 100%

Figure 4. SecSens Simulation Tool

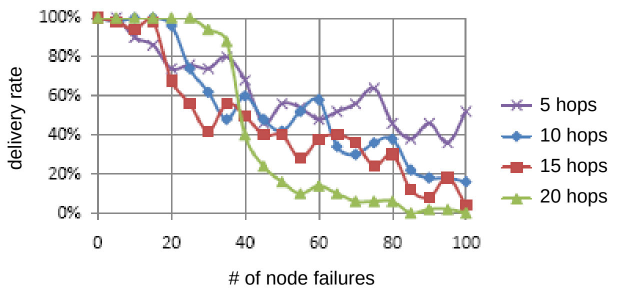


Figure 5. Message delivery rate after DoS attacks

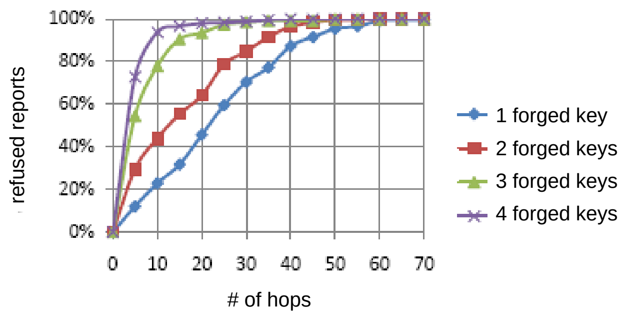


Figure 6. Rate of refused reports in reference to distance

verification keys (see previous section). Alternatively, he can forge missing keys. Figure 6 illustrates the rate of detected false reports in dependance of the distance between sender and receiver. As you can see in the results, the detection rate increases with more forged keys. Already after ten hops, SecSens can refuse %90 of false reports containing four forged keys.

The detection rate on intermediate nodes is directly dependent on the size N of the global key pool and number k of locally stored en-route keys (see previous section). With increasing number of en-route keys, the probability of an intermediate node to hold the same en-route key rises. Figure 7 shows the simulation results for using different numbers of en-route keys. As expected, the detection rate for false reports rises with increasing number of locally stored keys. On the other hand, the ratio k/N should not be too high, since a compromised node would irrevocably disclose a part of the global key pool.

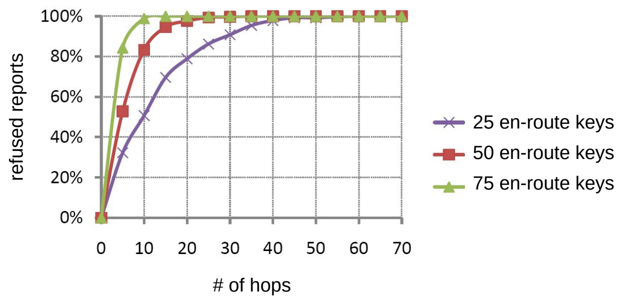


Figure 7. Influence of local stored en-route keys

In order to show the feasibility of SecSens on real environments, we established a testbed with different kinds of sensor nodes: ESB 430/1 and MSB-430 of Freie University Berlin (see Figure 8). Both sensor boards have the TI MSP430 microcontroller.

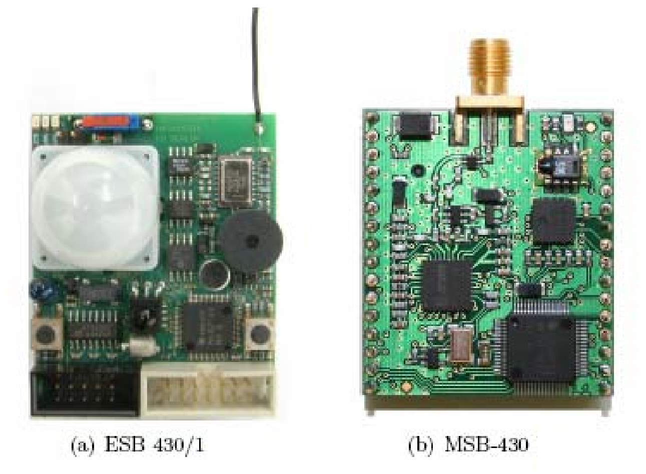


Figure 8. Sensor boards: ESB 430/1 and MSB-430

ESB 430/1 contains 60 KB flash memory and 2 KB RAM, whereas MSB-430 has 55 KB flash memory and 5 KB RAM. Because of the larger RAM, MSB-430 was used as cluster-head. We used four cluster-heads managing each four sensor nodes that were all ESB boards, making 20 nodes alltogether. Two PCs act as base stations. We could successfully perform all stages of SecSens which were described before.

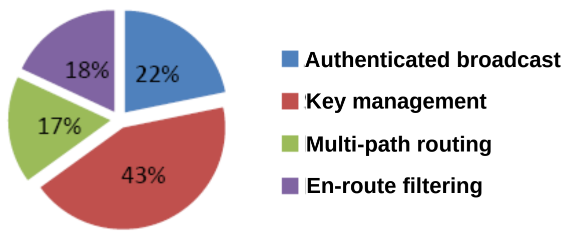


Figure 9. Memory map of SecSens security components

In order to use memory effectively, SecSens stores frequently changing data in RAM and relatively static data in EEPROM. Table I describes memory consumption of cluster-heads and sensor nodes. The most memory space is used for key management. Figure 9 shows the memory map of SecSens security components.

In the initial phase, energy consumption is comparatively high. For establishing the network and distribution of keys, cluster-heads consume in average $2.6Ws$ energy, whereas sensor nodes need only $0.2Ws$ (see Table II).

Energy consumption for sending reports depends on the distance between cluster-head and the base station. We established in a second test a network with up to 20 cluster-heads placed in a line side by side. The last cluster-head received level 20, which means that it needs 20 hops to reach the base station. We measured the energy consumption for sending reports from different levels. The multi-path routing ensures a robust transmission, but sending duplicated

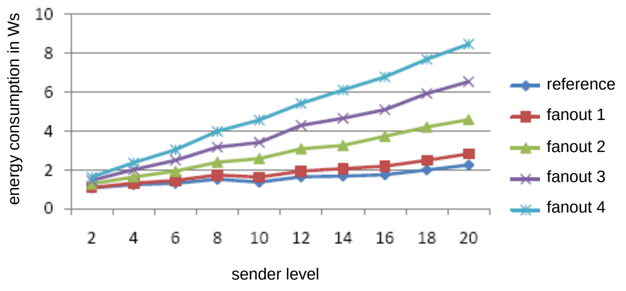


Figure 10. Consumed energy for reporting

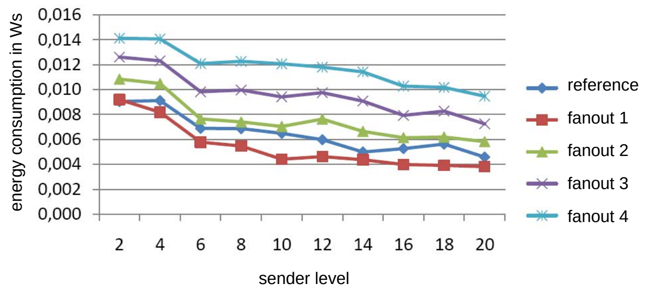


Figure 11. Energy consumption of single cluster head in reference to sender level

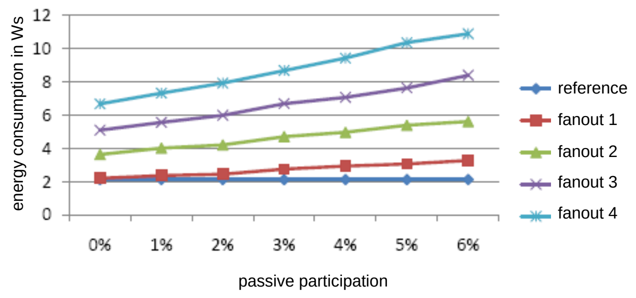


Figure 12. Energy consumption with passive participation

	packets	data	energy
sensor node	15	0,2 KB	0,2 Ws
cluster-head	142	4,1 KB	2,6 Ws

Table II
COSTS IN INITIAL PHASE

	AB	KM	MPR	EF	total
cluster-head: RAM	434 B	9 B	710 B	41 B	1194 B
cluster-head: EEPROM	800 B	2440 B	252 B	981 B	4473 B
sensor node: RAM	54 B	9 B	0 B	0 B	63 B
sensor node: EEPROM	0 B	92 B	0 B	537 B	629 B

Table I
SECSENS MEMORY REQUIREMENTS, (AB) - AUTHENTICATED BROADCASTS, (KM) - KEY MANAGEMENT, (MPR) - MULTI-PATH ROUTING, (EF) - EN-ROUTE FILTERING

packets from several routes (fanout) increases the energy consumption. Figure 10 shows consumed total energy in the sensor network for reports using no fanout (reference) or multiple fanouts. On figure 11 you can see the average energy consumption of a single cluster head in reference to the sender level. It is interesting to mention that energy consumption for sending four packets increases by only %200 instead of %400 as would be expected. The reason lies in the good balance of load in sending multi-path messages.

As mentioned above, nodes can listen to packet transmission of neighbors and can take the responsibility for forwarding with a certain probability in case of detected failures. Figure 12 describes the energy results for passive participation with different probabilities. Increasing passive participation naturally leads to higher energy consumption, because of the additional message delivery.

V. CONCLUSION

This paper presented *SecSens*, a multi-level security architecture for wireless sensor networks. SecSens combines several security approaches on different system levels in order to provide high protection. SecSens contains four components, which interact with each other: authenticated broadcasts, key management, routing, and en-route filtering. We implemented a simulation tool, where huge network sizes can be established. Evaluation results show that SecSens can resist DoS and insider attacks limiting failures to a local area. Infiltrated false sensor data can be refused with a high probability. We also demonstrated the feasibility of SecSens by building a real sensor network environment with two different kinds of sensor boards.

REFERENCES

[1] Faruk Bagci, Theo Ungerer, and Nader Bagherzadeh. SecSens - Security Architecture for Wireless Sensor Networks. In *The Third International Conference on Sensor Technologies and Applications (SENSORCOMM '09)*, Athens, Greece, June 2009.

[2] Haowen Chan, Adrian Perrig, and Dawn Song. Key distribution techniques for sensor networks. pages 277–303, 2004.

[3] Jing Deng, Richard Han, and Shivakant Mishra. Insens: Intrusion-tolerant routing for wireless sensor networks. *Computer Communications*, 29(2):216–230, 2006.

[4] P. Downey and R. Cardell-Oliver. Evaluating the Impact of Limited Resource on the Performance of Flooding in Wireless Sensor Networks. In *Proceedings of the 2004 international Conference on Dependable Systems and Networks*, Washington, DC, USA, June 2004.

[5] Chris Karlof, Yaping Li, and Joe Polastre. Arrive: Algorithm for robust routing in volatile environments. Technical Report UCB/CSD-03-1233, University of California at Berkeley, May 2002.

[6] Donggang Liu and Peng Ning. Multilevel μtesla: Broadcast authentication for distributed sensor networks. *Trans. on Embedded Computing Sys.*, 3(4):800–836, 2004.

[7] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, USA, September 2002.

[8] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. Spins: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002.

[9] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[10] R. Verdone and C. Buratti. Modelling for Wireless Sensor Network Protocol Design. In *International Workshop on Wireless Ad-hoc Networks (IWWAN 2005)*, London, United Kingdom, May 2005.

[11] Hao Yang, Fan Ye, Yuan Yuan, Songwu Lu, and William Arbaugh. Toward resilient security in wireless sensor networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 34–45, New York, NY, USA, 2005. ACM Press.

[12] Fan Ye, Haiyun Luo, Songwu Lu, and Lixia Zhang. Statistical en-route filtering of injected false data in sensor networks. *IEEE Journal on Selected Areas in Communications, Special Issue on Self-organizing Distributed Collaborative Sensor Networks*, 23(4):839–850, April 2005.

[13] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. Leap: efficient security mechanisms for large-scale distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 62–72, New York, NY, USA, 2003. ACM Press.

[14] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, and Peng Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. *2004 IEEE Symposium on Security and Privacy*, 00:259–271, 2004.