Thwarting Attacks on ZigBee – Removal of the KillerBee Stinger

Björn Stelte and Gabi Dreo Rodosek
Universität der Bundeswehr München
Faculty of Computer Science
D-85577 Neubiberg, Germany
Email: {bjoern.stelte,gabi.dreo}@unibw.de

***Abstract*—Wireless Sensor Networks (WSNs) have recently emerged as an important research topic. Due to the enormous number of sensor nodes and the constrained resources, specific research challenges can be identified with respect to security. Almost all available commercial and research sensor nodes are equipped with ZigBee transceiver chips, and thus making ZigBee the de-facto standard in WSN communication. Since Joshua Wright's KillerBee Framework was released with its focus on exploring and exploiting the security of ZigBee networks, non security-hardened WSNs increase the risk of being vulnerable against certain attacks such as simple association flooding and packet replay attacks. We propose an anomaly-based approach intrusion detection system (IDS) optimized for ZigBee-based WSN to protect ZigBee-based WSN nodes against KillerBee supported attacks. We describe the KillerBee attack procedure and propose an approach of guarding a ZigBee transceiver. Based on an extended sensor node/network simulation and analysis framework, we demonstrate furthermore how our anomaly-based detection engine can thwart attacks on a ZigBee transceiver.**

I. Introduction

Several attacks on WSNs are already known, including multiple attacks migrated from the TCP/IP world to WSNs. Attacks on WSNs can be classified in several categories: (i) sybil, (ii) wormhole, (iii) DoS, (iv) node replication, and (v) node compromise [1]. The resources of sensor nodes in a WSN are limited, thus a detection of attacks on the WSN communication infrastructure is a non-easy task.

An intrusion detection system (IDS) which monitors all sensor nodes and gives controller and gateway nodes enough information to build a kind of an early warning system is a needed but challenging task. The idea is that sensor nodes monitor their neighborhood, detect and report abnormal network traffic, and respond by certain countermeasures like isolation of malicious nodes in the network. A known attack which was discussed for some time is the sybil attack (e.g. [2]). This kind of attack enables hackers to influence routing, information exchange, etc. by generating multiple legal messages having different sources by one node with virtually multiple identities in the network [3]. In special situations, this kind of attack can be detected in WSNs as Lee et al. have shown in [3] by a method based on a challenge-response approach for tree-based sensor topologies. Nevertheless, a common IDS for WSNs is not part of any commercial communication standard used in WSNs like ZigBee or Bluetooth. With only constrained resources available, it is hard to detect malicious traffic which is, however, needed to build trustworthy and reliable WSNs. Secure WSNs are not only needed in military scenarios (battlefield surveillance, RF mine field, tracking objects) but also in healthcare or monitoring of critical infrastructures.

In this paper we assume that all sensor nodes are equipped with ZigBee transceivers. Since nearly all today's sensor platforms (XBow[4] MicaZ, IRIS, or TelosB) use ZigBee transceiver chips like the famous Chipcon CC2420 [5] this assumption is not far from reality. Currently, ZigBee is the de-facto industrial standard for small monitoring devices, deployable for simple monitoring scenarios up to critical infrastructure surveillance with hard security requirements. An excellent example for a critical ZigBee-based application is the smart energy monitoring approach. The U.S. Federal Energy Policy Act of 2005, California's Title 24 and similar initiatives across Europe identify requirements for smart energy metering and end-to-end demand response systems that needs to be implemented across the power grid. In the U.S. smart meters based on ZigBee technology are already enrolled widely (like the Ember ZigBee platform for home area networks). Other market fields for ZigBee controlled devices are health care, building automation, home automation, remote control, etc., but also military and boarder control units.

Since Joshua Wright presented the KillerBee framework at the ToorCon11 security conference, real world attack tools for attacking ZigBee networks are available. So, an efficient detection of attacks on the ZigBee networks within the WSN is currently of high importance. Network devices have to cooperate in terms of detection and classification of malicious network traffic, in order to build a kind of distributed IDS where several nodes need to be selected to protect and monitor parts of the network. Each IDS node may only be used for protecting nodes in a certain range (e.g., *k*-hop neighbors), and each node in the network needs k active IDS nodes to be protected. The efficient selection of IDS nodes among all nodes aims to minimize the number of IDS nodes while maintaining the *k*-protection for all nodes (including IDS nodes themselves). This problem is the same as the *k*-self-protection problem presented by Wang et al. [6]. IDS nodes are just the "active" sensors, and other non-IDS nodes are the "nonactive" sensors, as described in Section 2.

We propose a first anomaly-based IDS optimized for ZigBee-based WSNs, as far as we know [7]. We describe the

ISBN 978-3-901882-53-1, 9th CNSM and Workshops ©2013 IFIP

KillerBee framework, identify IEEE 802.15.4 / ZigBee security weaknesses, and present an approach where an adapted ZigBee transceiver could guard itself and help neighbors by cooperation in terms of self-healing and self-protection. We have implemented our detection algorithm within a simulation and analysis framework. The simulation results show that our anomaly-based approach can detect KillerBee attacks and protect the sensor nodes on further attacks.

In Section 2 the self-protection problem for WSNs is discussed. The ZigBee standard, its security features, and known security vulnerable are presented in Section 3. Some of the described security weaknesses are used by the KillerBee attack framework. This framework is in focus of Section 4 where we analyze two attacks of the framework in detail. Our approach of a WSN anomaly-based intrusion detection system is introduced in Section 5 followed by a description of our proof-of-concept implementation in Section 6. Section 7 finally concludes the paper.

II. Self-Protection for WSN

Wang et al. first formally presented the general self-protection problem in WSNs ([6]). The problem focuses on providing protection to sensor nodes themselves instead of monitoring e.g. critical infrastructure in certain surveillance scenarios. The idea is that sensor nodes of a WSN can resist attacks targeting on them directly. It was defined that a WSN node is *k*-self-protected if there are at least *k* active sensor nodes that can monitor the node (active or sleeping) at any moment. Thus, the self-protection problem is different to the fault-tolerance problem. Fault tolerance focuses on providing high connectivity of the network while self-protection does not address connectivity issues. Energy consumption is always a major concern in WSNs, therefore Wang et al. defined the following two energy measures.

Definition A Minimum Self-Protection is a self-protection approach for a sensor network where the number of nodes selected to be active is minimized at a certain time.

Definition A Maximum Disjoint Self-Protection is a set of disjoint self-protections for the sensor network, where the cardinality of the set is maximized.

Wang et al. proved that both the minimum self-protection problem and the maximum disjoint self-protection problem are NP-complete, and gave a centralized method with O(log n) approximation ratio, where n is the total number of sensors. Efficient centralized and distributed algorithms for a minimum self-protection problem with either a homogeneous or heterogeneous sensing radius are known (e.g. [6], [8]).

In [8] the presented centralized and distributed methods show that it is possible to find a *k*-self-protection set whose size is within at most 10 times of the optimum when the sensing ranges of all sensors are uniform. The provided methods can find a *k*-self-protection set with approximation ratio $\mathcal{O}(\log_2 \gamma)$ when sensing ranges are heterogeneous (γ is the ratio of the maximum sensing range over the minimum sensing range).

III. ZigBee Standard

Among the latest events of the wireless revolution, the increasing importance of ZigBee as a standard for WSN is certainly one of these. ZigBee and IEEE 802.15.4 had been proving in the last years that they can achieve the same results as WiFi had achieved for high bit-rate wireless LANs and some large reliable deployments are now in place implementing ad-hoc WSN in critical applications, like environment monitoring, asset tracking, and also military scenarios [9].

The first release of the ZigBee standard was in 2004, followed by two revisions in 2006 and 2007 (ZigBee 2006v and ZigBee-Pro 2007v). The main differences between the first ZigBee revision and the latest ZigBee-Pro release are stochastic addressing, mesh data management, packet fragmentation, dynamic best channel choice, optional asymmetric connections and security improvements. Improvements concerning the security are made by optional provided header and/or data cryptography with AES-128. In ZigBee-2006 a global network key is used to create secure communication. ZigBee-Pro offers a more complex system which adds a peer-to-peer encryption layer. Each couple of nodes has its own key which allows a peer-to-peer encryption. However, we will focus on the ZigBee standard since ZigBee-Pro is today almost not used on WSN platforms.

Currently, all relevant WSN nodes, like MicaZ, IRIS, TelosB, etc., use ZigBee-conform transceivers to communicate with each other. The ZigBee standard for short-range wireless networking is targeted mainly for battery-powered applications where low costs and low power consumption are the main requirements. Technically, the ZigBee standard is based on IEEE 802.15.4 PHY/MAC standard. The ZigBee Alliance is only promoting technology, like stack protocol development, interoperability guarantee and application profile, and certification for home automation, industry automation, automatic metering infrastructure, telecommunication value-added services, personal health care, etc.

The IEEE 802.15.4 MAC layer implements several features which are used by the ZigBee protocol in network and application layers. The security services are one of these features. The underlying IEEE 802.15.4 protocol defines the encryption algorithm to use when transmitted data should be cyphered. However, IEEE 802.15.4 does not specify the key management or what kind of authentication policies have to be applied. These issues are treated in the upper layers which are managed by protocols such as ZigBee. The ZigBee standard implements two extra security layers on top of the IEEE802.15.4 protocol, namely the network and the application security layer.

In ZigBee the AES cipher algorithm with a 128 Bit key length is used for encryption and decryption. The encryption is only optional available, and need to be activated explicitly. Concerning the key management, ZigBee knows three different type of keys:

- **Master Key** Master keys are pre-installed on each node. Their function is to keep the network keys exchange between two nodes in the key establishment procedure

confidential.

- **Network Key** It is used to cypher all data sent within the network. It is a unique 128 bit key shared among all devices in the network. It is generated by a Trust Center (TC) and regenerated at different intervals. Each node has to get the network key in order to join the network. Once the TC decides to change the network key, the new one is spread through the network using the old network key.
- **Link Key** It is used to cypher data at the application layer and to send the 'network key' cyphered. It is unique between each pair of nodes. This key is not configurable although it must be specified if it is going to be used or not. The keys are managed by the application level and are used to encrypt all information that is exchanged between each two devices. Due to high memory usage, this kind of key is mostly unusable for WSN applications.

ZigBee networks need at least one coordinator (personal area network, PAN). If encryption is enabled on the coordinator, a security policy is applied to the network upon creation. This means that e.g. enabling security adds an authentication step to the joining process. For instance, after a router joins a network, it must obtain the network security key to become authenticated. If the device cannot obtain the network security key, authentication fails and the device leaves the network since it cannot communicate with anyone on the network.

Additionally to the coordinator, a TC is needed. Only a coordinator can overtake the part as a TC. The TC is a single device that is responsible for determining who may join the network. If a TC is enabled, it must approve each router or end device which want to join the network. If a router allows a new device to join the network, the router sends a notification to the TC that a join has occurred. The TC instructs the router to either authenticate the newly joined device or to force the device to leave. A TC is required for some public ZigBee profiles where cryptography is activated. To use the TC in a ZigBee network, the coordinator should set the "use TC" bit correctly in the 'Encryption Options' parameter before starting a network. As mentioned before, only the coordinator can serve as a TC. Therefore, in large ZigBee networks often the coordinators of the network and the TCs are implemented on the same node due to scalability and performance reasons.

Modules define a network key and a link key (TC link key). Both keys are 128 bits and are used to apply AES encryption to RF packets. The coordinator selects a network security key using the 'Encryption Key' parameter. Similarly, the coordinator must also specify a link key using the 'Link Key' parameter. Each pair of devices can have set both network and link keys. In this case the link key is always used. There are two kinds of security policies which the TC can follow:

- **Commercial Mode** The TC share master and link keys with any of the devices in the network. This mode requires high memory usage. This mode offers a complete centralized model for the key security control.
- **Residential Mode** The TC shares just the network key (it is the ideal mode when embedded devices have to cope with this task due to the low resources they have). This is the mode normally chosen for the WSN model (Figure 1).

When a new device wants to join a secured (encrypted) network, it must obtain the network key from the coordinator in order to send data. The coordinator will either transmit the network key in a clear form, or it can encrypt it using a pre-installed link key. If the 'Encryption Options' bit is set to transmit the network key unencrypted or if the 'Link Key' parameter is set to 0 on the coordinator (select a random link key), the coordinator will transmit the network key unencrypted. Otherwise, if the 'Encryption Options' bit is not set and the 'Link Key' is greater than 0, the coordinator will encrypt the network key with the link key and transmit the network key encrypted to any joining devices. It has to be noticed that association messages are always send unencrypted, and that only associated nodes are able to join a secured network.

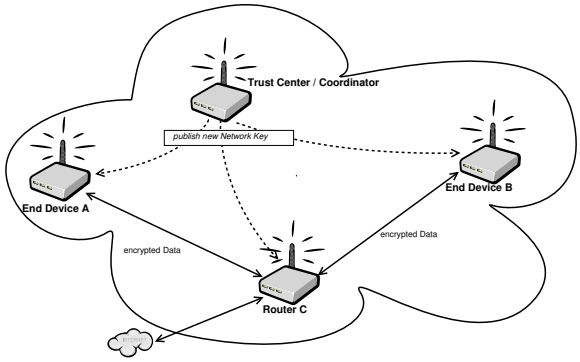

Fig. 1. ZigBee Residential Mode

IEEE 802.15.4 defines security policies for data confidentiality, data authenticity, and replay protection. If a joining device does not have the right pre-configured link key, and the network key is being sent encrypted, then the joining device will not be able to join the network. Network security requires a 32 bit frame counter to be maintained by each device. This frame counter is incremented after each transmission and cannot wrap to 0. If a neighbor receives a transmission with a frame counter that is less than or equal to the last received frame counter, the packet will be discarded.

A 4-octet frame counter is include in each IEEE 802.15.4 secured frame and used to provide replay protection. It is also needed for semantic security of the cryptographic building block used for securing outgoing frames[10]. As described in the standard, the frame counter is incremented each time an outgoing frame is secured. If reaching the frame counter's maximum value (0xFFFFFFFF), the associated keying material can no longer be used. To prevent an eventual lockup where the frame counter on a device reaches 0xFFFFFFFF, the network key should be periodically updated on all devices in the network. To update the network key in the network, the coordinator should issue the 'Encryption Key' parameter with a new security key. This will send a broadcast retransmission

throughout the network causing that the frame counters on all devices are reset to 0. With this, the devices begin to use the new network key. All devices will also retain the previous key for a short time until everyone has switched to the new key.

Nevertheless, these security policies will not efficiently prevent to jam the association process or to execute a replay message attack. The following security problems are known:

- **Weak integrity protection on AES-CTR** Integrity protection is based on a simple CRC calculation. It is possible to change the payload and then recalculate the new CRC. With this it is possible to forge messages to begin confidentially attacks.
- **DoS Attack on AES-CTR** As a sequential freshness mode is used, a unique forged packet with the frame counter and key sequential counter set to the maximum value will stop receiving any other frame from this address.
- **Non effective packet replay protection** IEEE 802.15.4/ZigBee has only meager replay protection (4-octet frame counter).
- **Device association requests are send in clear** ZigBee does not use a challenge-response process nor signatures to at least secure basically the association process.
- **More and more functionality is done by hardware (ZigBee transceiver)** Software is flexible, hardware is not. A bug in the hardware implementation cannot be easily corrected.
- **CCMP known plaintext recovery** ZigBee has problems with CCMP as a stream cipher and IV reuse as any stream cipher has potential IV reuse issues.
- **By default NO security** The application controls the security required, ACK packets are always send unencrypted, other packets can optionally use encryption or integrity checks.
- **Examples of IEEE 802.15.4 MAC layer attacks:**
  - Guaranteed Time Slot (GTS) attack [11]
  - Back-off interval manipulation (DCF) [12]
  - ACK attack [13]
  - PANId conflict attack [14]
  - . . . and more [1].

These theoretical attacks show that the ZigBee protocol and its implementations can be compromised. Mostly, application developer do not activate all possible security related options in their sensor application. Therefore, it is in most cases easily possible to attack a WSN even with less complex attack methods. In the next section we present a free available ZigBee attack framework that use these security problems. The framework is a proof-of-concept enabling "bad guys" to easily attack deployed ZigBee networks.

IV. KillerBee Attack Framework

Joshua Wright has developed a framework attacking ZigBee and IEEE 802.15.4 networks called KillerBee[15]. This framework simplifies sniffing and injecting network traffic, next to packet decoding and manipulation. So far, two *AVR RZ Raven* USB sticks are needed to physically attack a deployed ZigBee network. The framework provides a modified firmware for the USB ZigBee transceiver sticks. Once uploaded to the sticks the user has full control over the USB stick, and can sniff and send injected ZigBee packets. The framework includes the following tools (as proposed by Joshua Wright in [15]) and assumes that security features are not activated:

- **zbassocflood:** Repeatedly associate to the target PANId in an effort to cause the device to crash from too many connected stations.
- **zbconvert:** Convert a packet capture from libpcap to Daintree SNA format, or vice-versa.
- **zbdsniff:** Captures ZigBee traffic, looking for NWK frames and over-the-air key provisioning. When a key is found, zbdsniff prints the key to stdout.
- **zbdump:** A tcpdump-like took to capture IEEE 802.15.4 frames to a libpcap or Daintree SNA packet capture file. Does not display real-time stats like tcpdump when not writing to a file.
- **zbfind:** A GTK GUI application for tracking the location of an IEEE 802.15.4 transmitter by measuring RSSI. Zbfind can be passive in discovery (only listen for packets) or it can be active by sending beacon request frames and recording the responses from ZigBee routers and coordinators.
- **zbgoodfind:** Implements a key search function using an encrypted packet capture and memory dump from a legitimate ZigBee or IEEE 802.15.4 device. This tool accompanies Travis Goodspeed's GoodFET hardware attack tool, or other binary data that could contain encryption key information such as bus sniffing with legacy chips (such as the CC2420).
- **zbid:** Identifies available interfaces that can be used by KillerBee and associated tools.
- **zbreplay:** Implements a replay attack, reading from a specified Daintree DCF or libpcap packet capture file, retransmitting the frames. ACK frames are not retransmitted.
- **zbstumbler:** Active ZigBee and IEEE 802.15.4 network discovery tool. Zbstumbler sends beacon request frames out while channel hopping, recording and displaying summarized information about discovered devices.

Two of these tools are of specific interest, *zbassocflood* and *zbreplay*. Joining a ZigBee network is possible by two ways: MAC association and network (NWK) rejoin. The later is possible if a node already knows the correct and actual NWK crypto key. Therefore, only nodes who have disassociate themselves once and now try to rejoin the network can use this mode. Nodes trying to join a network for the first time have to use the association method. The coordinator send periodically beacon messages which a device receives and internally decides if it wants to join the network or not. A joining device send an *Association Request* Message to the Coordinator (Short-Address: 0x0000) and waits for an acknowledgment. In a second message the joining device

transmits a *Data Request* command to the coordinator. Also this transmission is acknowledged by the coordinator. In the next phase the Coordinator sends an *Association Response* message to the device which the device answers by an acknowledgment message with the sequence number of the *Association Response*. The *Association Response* message includes a short address for the device to use while associated to the ZigBee network. Since all the associated frames are sent unencrypted, the MAC association procedure is vulnerable to packet manipulation. The KillerBee *zbassocflood* uses exactly this vulnerable. Therefore, the tool produces randomized MAC addresses and starts to associate non-existent devices. Since every associated device gets an unique short address (16 Bit), the amount of available address is limited and the KillerBee tool can very quickly simulate a huge amount of joining devices. A ZigBee transceiver has to hold a kind of address resolution table for conversion of short and long (MAC) ZigBee addresses. A coordinator can control up to 2^{16} devices (address space of short local addresses). Thus, a transceiver willing to cope with 2^{16} addresses has to store an address resolution table of $16 * 2^{16}Bit + 64 * 2^{16}Bit$ (640 kByte) size. This amount does not seem to be huge but a typical microcontroller on a sensor node does not have 640 kByte free memory nor has the transceiver such an amount of memory. By the way, a normal ZigBee stack is about 120 kByte size in total. A KillerBee association flooding attack could be - depended on the transceiver implementation - harmful even if not the whole address space is blocked. Therefore, an association flooding has to be detected early enough otherwise the transceiver will not allow new associations or even run out of memory and crash. As a result, the ZigBee coordinator node could get unavailable (DoS attack) and all associated nodes will be truncated from the environment till joining another network.

The second tool called *zbreplay* attacks the IEEE 802.15.4 protocol by re-sending former sniffed packets. Joshua Wright compares this attack with the well-known ARP attacks of former days [15]. The given example shows how successful and harmful such an attack could be if one considers that for example replaying a message actuating water control valve to open one degree several times. As mentioned before, IEEE 802.15.4 specifies frame counters and at least in ZigBee-Pro transceivers also freshness checking (time-based protection). The problem is that replay protection can be turned on or off, the application developer has to explicit turn this feature on to activate the build-in replay protection mechanism. Many WSN applications are built with optimization in mind, especially concerning power consumption. As a consequence, developers often try to reduce the lines of code and also the amount of additional used features to a minimum.

Replay protection attacks are well-known for which the IEEE 802.15.4 specification specifies replay protection mechanism. The protection mechanism is used to accept a frame by checking whether the frame counter of the recent message is larger than the previous one. However, if an adversary sends many frames with large counters to a legitimate node, ZigBee nodes using the replay protection mechanism will reject the legitimate frames with small counters from other nodes.

If a IEEE 802.15.4/ZigBee network is properly configured with respect to security features, it is not easy to be hacked at least not by KillerBee supported attacks. However, the activation of security features in ZigBee is a non-trivial task, and most available application neglect the security options.

V. WSN Intrusion Detection

While IDSs in traditional networks are widely spread, including a large number of research work, intrusion detection in wireless networks is still in its infancy [7]. Intrusion detection in wireless networks faces several challenges as for example:

- **Non-continuity:** Because of the limited power resources, the sensor nodes are forced to go to a sleep mode as often as possible to maximize the period of application. As a consequence, a permanent surveillance of the traffic is not possible.
- **Numerous potential attacks and endangerment:** In addition to traditional attacks wireless sensor nodes are forced to various attacks, e.g. kinds of replay-attacks, spoofing, isolation and deception. In this context, also the environment has to be considered, such as sensor nodes can loose communication because of the surroundings or weather conditions or being isolated by external manipulations.
- **Highly dynamical network topology:** The sensor nodes and therefore the network topology are typically highly dynamical due to changes in the environment.
- **Limited resources:** The computational and memory capabilities of the sensor nodes are extremely limited, e.g. an AtMega128l processor runs at 4 MHz and has 128 KB program memory.
- **Incompleteness:** The possible drop out of nodes must be kept in mind, e.g. because of isolation by weather conditions or by damage of a sensor node.

Therefore, an IDS for WSNs must fulfill the following two main tasks:

First, the system must be designed and realized in a **lightweight implementation**: The code must be minimized regarding its line of codes, and especially with respect to memory and processing power consumption.

Secondly, the detection engine must be **anomaly-based**: A payload analysis on the sensor nodes is not possible because of the needed energy and computational power. Furthermore, the payload analysis strongly depends on the up-to-dateness of the corresponding pattern database. With respect to sensor nodes, this also demands a very high amount of memory space. For example, the *Snort* database of intrusion patterns already has a compressed size of nearly 90 MB containing 15000 attack patterns. This can be optimized with respect to the possible attacks when using sensor nodes but additional attacks for WSNs have to be added. As a consequence, it is impossible to use a signature-based system on a sensor node.

Besides, a pattern-based system only can be effective if the database is always up-to-date which requires regularly updates.

This can not be guaranteed especially in the environment of an operational WSN.

Therefore, it is not possible to use a pattern-based approach neither by means of disk usage or the energy consumption nor by the needed patterns itself.

An anomaly-based system has to build a model of the normal network behavior in advance. Afterwards, the state of the system is measured and compared to the expectation of the model. If the result exceeds a specific threshold, an alarm is raised. Because of that, a database is not needed and a deployment on the sensor nodes is possible.

One of the most challenging tasks of anomaly-based systems resulting from its functional principle is the necessity of a learning phase to build the needed behavioral model. Especially sensor networks are deployed typically in a hostile environment, therefore a learning phase is not acceptable: If there are attacks or malicious behavior in the environment right from the beginning of the network usage, the malicious behavior will be learned as a normal one and no alarm will be raised after the learning phase for the malicious events. Also, because of the dynamic characteristics of the sensor network, the features for the intrusion detection must be based on parameters which are independent from the specific structure of the wireless network to a specific time, otherwise a definition of the normal network behavior is not possible.

As a consequence, the detection must be based on the typical communication behavior of single sensor nodes and on environmental-independent parameters of the inter-node communication. Of course, these features must be selected in a way that a reliable detection of attacks is still possible. Thus, it is necessary to analyze the task and functionality of a single node, and than the expectation of the communication profile is being evaluated which can be modeled in a simulation and afterward used for the training of the model. By detaching from the specific environmental conditions and focusing on key features of the sensor communication, the behavior-model can be built, and the learning phase can be fulfilled in advance to the deployment.

Based on these requirements, the following features are useable in a WSN for an anomaly-based detection:

- Number of neighbors (active, overall)
- Communication with neighbors (active, observed, none)
- Number of connections (ingoing respectively outgoing)
- Number of message packets, length, bytes per packet
- Number of data requests
- Connection times
- Administrative and operational network messages

These features are mainly from a static nature in a sense, that the coarse parameters will be in specific ranges for the whole deployment, therefore trainable in advance. Furthermore, features with a more dynamic character can be included if a training is possible before setting up the sensors in the area of operation, and an adaption of the model is possible during the operation and within defined boundaries.

Based on these features, three detection vectors can be defined: (i) local node, (ii) local area and (iii) network-wide evaluation. As a consequence, the architecture of the IDS for WSNs consists of three separate calculable normality models, namely the *sensor-specific*, the *regional communication* and the *network communication* behavior model. The third vector is mainly relevant for the coordinator while the first ones are evaluated on every sensor node: Each node monitors the parameters for the own communications, therefore being able to detect anomalous behavior which targets the node himself. By analyzing the communication between other sensor nodes within the radio range, also attacks targeted onto other nodes can be detected. This information can be sent to the coordinator or other nodes in the neighborhood to realize an distributed IDS for the sensor network. If a sensor detects a misbehaving node in his neighborhood, the node will be ignored in the further communication process. Nevertheless, it is possible to include the node again in the communication if the behavior regains normal to a later point in time or to isolate the node for the rest of the WSN deployment. At the moment, only the sensor-specific detection is implemented while the other vectors will be included in the next prototype of our IDS.

For the first proof-of-concept, a minimized and lightweight IDS for the sensor nodes was implemented. It will be advanced with the evaluation of additional statistical parameters in the future with the focus of an energy-efficient implementation using minimal resources. At the moment, the short and long addresses of the conversations are monitored and evaluated. The parameters have to be trained to the specific safeguarded network. This is done in the learning phase which is shown in principle in Algorithm 1. As mentioned before, it is possible to carry out this phase in advance in a simulated and secure environment, therefore not endangering the learning process.

Algorithm 1 Learning phase

```
finish_time = get_time() + learning_time {calculate the finish time}
while get_time() < finish_time do
  receive_byte()
  if message_complete() then
    evaluate_address()
    update_hashtables() {update short- and long-address hashtables}
  end if
end while
write_environment() {save the parameters of the detected communications}
set_thresholds() {calculate and set the thresholds}
```

For conserving the resources, an evaluation is done at periodic points in time, chosen by the parameter `evaluation_step`. The variations in the addresses involved in the communications as well as the number of packets are evaluated for the specified time window in our first prototype (see Algorithm 2).

Algorithm 2 Detection phase

```
evaluation_time = get_time() + evaluation_step
loop
  receive_byte()
  if message_complete() then
    evaluate_address()
    update_hashtables() {update short- and long-address hashtables}
    if get_time() > evaluation_time then
      if get_environment() + threshold < size(hashtable) then
        drop_communication() {intrusion detected: drop message}
      else
        update_environment()
        update_thresholds()
      end if
      evaluation_time = get_time() + evaluation_step
    end if
  end if
end loop
```

VI. Proof-of-Concept

For a proof-of-concept we have extended, the *avrora* simulation and analyze framework [16]. Avrora is a instruction-level sensor network simulator for simulating WSNs of motes like the Xbow Mica-series. The latest version of avrora simulates XBow MicaZ and TelosB motes, both are equipped with the Chipcon CC2420 ZigBee transceiver. Therefore, we use an avrora simulation for testing our attack detection approach by simulating a 16 node sensor network with one controller node and one misbehaving node. The misbehaving node starts to flood the network with association attempts, setting the MAC address to a random value for each attempt. The simulation itself executes example TinyOS images of the association example provided by the TinyOS 2.0 / Open-ZB sources [17].

Attacks on coordinator nodes within the simulation will start about 60 seconds after the simulation starts to give our algorithm enough time for a learning phase. This is realistic in such a way that we assume that a real network is already (at most) deployed and active.

We have implemented our detection and attack prevention code in the simulated Chipcon CC2420 transceiver module of the avrora simulator. So, we have extended the ZigBee transceiver by a method to detect association flooding and to detect packet replay attacks. A detected misbehaving node will be isolated, the controller node behaves in a self-healing manner.

Because the initial network can be provided for the learning phase, the thresholds can be set very low: Only a few number of nodes will join the sensor network over time. Therefore, also slow-driven attacks (for exhausting the resources, the attacker must send at least a minimum number of requests to prevent the coordinator from going into the sleep mode) are detectable.

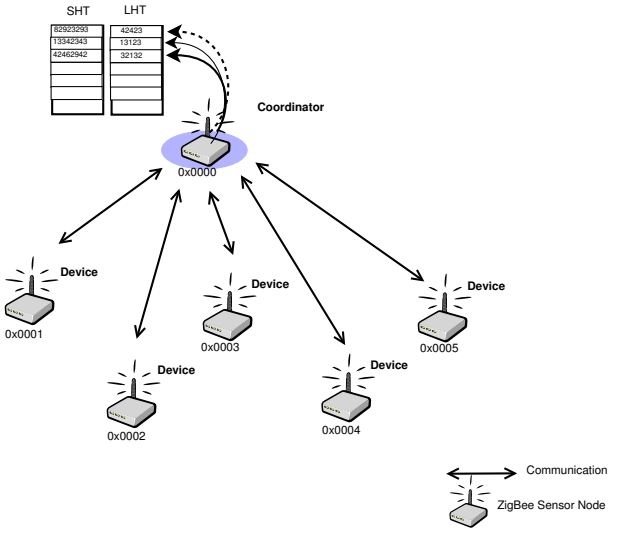

Fig. 2. Communication in the sensor network. Association requests are send to the coordinator. Hashes for the long- and short-addresses are saved into two hash-tables.

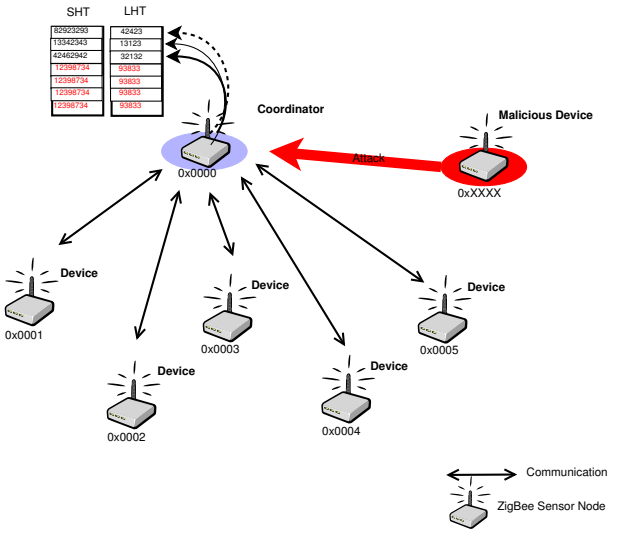

Fig. 3. The attacker sends faked association requests to the coordinator to prevent entering the sleep mode. New addresses are dropped into the hash-tables, filling up it quickly.

Figure 4 shows the evaluation of the MicaZ simulation results of the attack scenario. In the first 30 seconds the sensor network is observed and the parameters are trained. A very fast learning process is possible, because of the small size of the WSN that is reaching a steady communication soon. Anyway, in more complex networks, the learning phase can be extended. A period of normal network activities follows up. After that, a malicious node is activated which is starting an attack at second 63. Because of the high number of malicious association requests, the addresses seen in the network by the IDS are increasing quickly. Different values for the parameters like the window-size for the analysis and the evaluation-step have been used. A reliable detection of the attack was possible after $window-size[sec]+2.5sec$ in average, after the

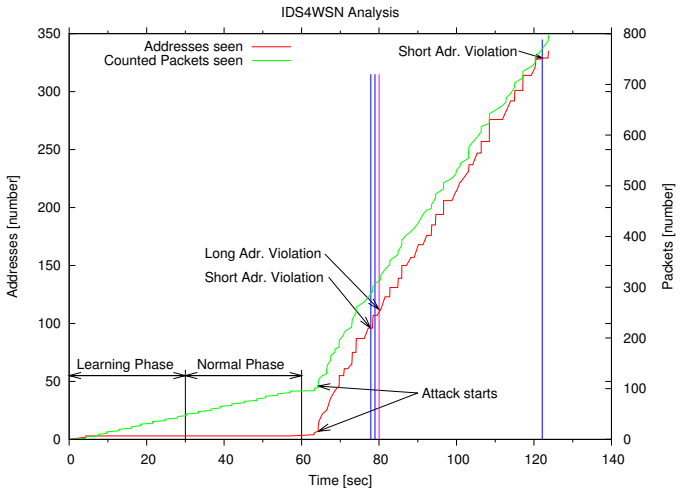

Fig. 4. Evaluation of the short- and long addresses and the counted packets in the IDS on the sensor nodes. Two window-sizes have been used for the analysis, 15 seconds and 60 seconds. For the small window, the first alarm is raised at second 77.8, for the bigger window the first alarm is raised at second 122.13. After the detection, further steps can be taken like dropping the packets for a specific time.

attack had been launched. The evaluation of the dependencies between the detection time, window-size, evaluation-step and the false alarm rates will be subject of our further work. After an alarm has been detected, the sensor node is able to take further steps like ignoring the malicious association requests.

VII. Conclusion

Today, sensor nodes use ZigBee transceivers to communicate with each other. Since the ZigBee standard is based on IEEE 802.15.4, attacks on IEEE 802.15.4 MAC layer are harmful to ZigBee-based devices. The presented KillerBee analysis shows that frame security of IEEE 802.5.4 MAC is a critical issue. Frame security is a set of optional services that may be provided by the IEEE 802.15.4 MAC to ZigBee. The problem is that if an application does not set any security parameters, then non security feature is enabled by default. Joschua Wright has implemented KillerBee to provide administrators the possibility to justify the added cost of enterprise-security ZigBee technology by hardware tamper-proof security features[15].

Regardless whether optional security features are activated or not, an attacker will try to find security holes. For this purpose, KillerBee is currently the one-of-a-kind ZigBee attack tool. We have analyzed the KillerBee framework and found two relevant attack mechanisms. Both attacks (association flooding and packet replay attack) can be found by an anomaly-based IDS. The challenge is to develop an IDS that meets the harsh requirements of WSNs. Our proposed approach and finally the proof-of-concept implementation shows that the proposed anomaly-based IDS fulfills these requirements.

Since an ambitious hacker can always start a denial-of-sleep attack with Josh's framework, we have implemented our IDS in the radio module. When an attack is detected, malicious packets are dropped and the micro-controller will not get an interrupt signal to wake-up and receive the packet. This kind of malicious traffic filtering is another important step towards more secure WSN nodes and to reduce the impact of denial-of-sleep attacks on sensor nodes.

Acknowledgment

The authors wish to thank the members of the Chair for Communication Systems and Internet Services at the Universität der Bundeswehr München, headed by Prof. Dr. Gabi Dreo Rodosek, for helpful discussions and valuable comments on previous versions of this paper.

This work was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

References

[1] D. R. Raymond and S. F. Midkiff, "Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses," *IEEE Pervasive Computing*, vol. 7, pp. 74–81, 2008.

[2] J. Douceur and J. S. Donath, "The Sybil Attack," in *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002, pp. 251–260.

[3] G. H. Lee, J. S. Lim, D. K. Kim, S. H. Yang, and M. H. Yoon, "An Approach to Mitigating Sybil Attack in Wireless Networks using ZigBee," in *ICACT*, Feb 2008.

[4] Crossbow, "Mica data sheet," June 2013. [Online]. Available: http://www.xbow.com/Products/Product_pdf_files/MICA%20data%20sheet.pdf

[5] Texas Instruments / Chipcon, "Cc2420 data sheet," June 2013. [Online]. Available: http://www.ti.com/lit/gpn/cc2420

[6] D. Wang, Q. Zhang, and J. Liu, "The self-protection problem in wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 3, no. 4, p. 20, 2007.

[7] M. Xie, S. Han, B. Tian, and S. Parvin, "Anomaly detection in wireless sensor networks: A survey," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1302–1325, 2011.

[8] D. Dong, X. Liao, Y. Liu, C. Shen, and X. Wang, "Edge Self-Monitoring for Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. PrePrints, 2010.

[9] X. Carcelle, B. Heil, C. Chatellier, and P. Pailler, "Next WSN applications using ZigBee," in *1st IEEE Home Networking Conference, Paris - France*, December 2007.

[10] P. Barontib, P. Pillaia, V. W. C. Chooka, S. Chessab, A. Gottab, and Y. F. Hu, "Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and ZigBee Standards," *Computer Communications*, vol. 30, no. 7, pp. 1655–1695, 2007.

[11] R. Sokullu, O. Dagdeviren, and I. Korkmaz, "On the IEEE 802.15.4 MAC Layer Attacks: GTS Attack," *Sensor Technologies and Applications, International Conference on*, vol. 0, pp. 673–678, 2008.

[12] S. Radosavac, A. A. Cárdenas, J. S. Baras, and G. V. Moustakides, "Detecting IEEE 802.11 MAC layer misbehavior in ad hoc networks: Robust strategies against individual and colluding attackers," *J. Comput. Secur.*, vol. 15, no. 1, pp. 103–128, 2007.

[13] Y. Xiao, S. Sethi, H. Chen, and B. Sun, "Security services and enhancements in the IEEE 802.15.4 wireless sensor networks," in *Proceedings of IEEE GLOBECOM*, vol. 3, 2005.

[14] R. Sokullu, I. Korkmaz, O. Dagdeviren, A. Mitseva, and N. Prasad, "An Investigation on IEEE 802.15.4MAC Layer Attacks," in *Proceedings of 8th International Symposium on Wireless Personal Multimedia Communications (WPMC 2005), Aalborg, Denmark*, Sep. 2007.

[15] J. Wright, "Killerbee:Practical Zigbee Exploitation Framework," in *Toorcon11*, Oct 2009.

[16] K. Mayoral, B. Titzer, and J. Palsberg, "Implementing MicaZ Support for the AVRora Simulator," 2005. [Online]. Available: http://research.cens.ucla.edu/pls/portal/url/item/0424D37AF398109CE0406180528D7106

[17] A. Cunha, A. Koubaa, R. Severino, and M. Alves, "Open-zb: an open-source implementation of the ieee 802.15.4/zigbee protocol stack on tinyos," *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, vol. 0, pp. 1–12, 2007.