Security in Wireless Sensor Networks:
an Overview

Murad Kamalov
murad@ut.ee
University of Tartu

1 Introduction

Recent breakthroughs in micro sensor technologies allow us to engineer compact, battery powered, wireless communication enabled and cheap sensing devices, sensor nodes. Combining those cheap sensor nodes into networks allows us to create Wireless Sensor Networks (WSN).

Figure 1 shows the general architecture of a typical WSN. Sensor nodes are usually low power, with limited computational and storage capabilities. Base station usually serves, for WSN, as a gateway to external network. Base station is typically more powerful device, than sensor nodes, with long lasting battery and better storage capabilities. Base station can communicate to external network (which is usually in the Internet) using wired and/or wireless communication channels.

The main goal of every WSN is to perform certain sensing tasks and report the results, in appropriate form, to the remote operator. WSN tasks are classified into three categories: space monitoring, monitoring assets (things), monitoring interactions [10].

− Space monitoring usually involves monitoring a certain environment. For example, in agriculture, climate control or surveillance.
− Monitoring things involves monitoring certain objects, for example health of a patient.
− Monitoring interactions involves such things as, asset tracking and manufacturing process flow.

Specifics of WSNs create many challenges, not only to the problem of security in WSNs, but also to the problems of efficient routing and data aggregation. Here are some of the WSN specifics. Firstly, WSNs typically work in uncontrolled environment, where attacker might have easy physical access to sensor nodes and thus compromise the operation of WSN. Secondly, sensor nodes communicate over wireless channels, which are very easy to eavesdrop and hence acquire confidential data. Thirdly, sensor nodes work on battery power, which means that communication protocols, security protocols, implementation of cryptographic primitives, data acquisition and processing algorithms should be optimized, in order for sensor nodes and WSN in general to operate reasonable amount of time. Fourthly, sensor nodes have limited computational capabilities, which significantly narrows down the choice of algorithms and security primitives for WSNs.
Fifthly, data in WSN is usually routed, not through routers, which end-host can usually rely on, but through similar battery powered sensor nodes, which means, that if neighbors of certain sensor node A become unreachable (drained battery or other reason), sensor node A (which still has some power) also becomes unreachable, because he was communicating with a base station through his neighbors. In addition, in some cases WSN can also become Delay-Tolerant Network (DTN), where transmission delays are very high and intermediate nodes should store aggregated data from multiple neighbour nodes.

All the challenges of WSNs, described above, make security in WSNs a very challenging problem, which requires in-depth security requirements/trade-off analysis of each particular application.

The rest of the paper is structured as follows. In section 2, we will go deeper into Security of Wireless Sensor Networks, general security requirements of WSNs will be defined and security challenges in WSNs will be explained. In section 3, we will go deeper into cryptographic primitives, used in WSNs, both asymmetric and symmetric cryptography based solutions will be presented. In section 4 we will review some of the existing implementations of security solutions used in WSNs. In particular, TinySec, ContikiSec and ZigBee security will be covered. Section 5 concludes the paper.

2 Security in Wireless Sensor Networks

There are three types of WSN architectures:

- Gateway Architecture, where sensor nodes communicate with external network through gateway. Usually base station is in a role of a gateway. The goal of a gateway is to translate protocol used in the WSN to TCP/IP used
in external network. In addition, gateway processes and aggregates the data received from sensor network. WSNs with gateway architecture are currently the most popular.

– Overlay Architecture, where overlay network is built on top of TCP/IP, used by external network and WSN transport layer protocol, used in WSN. The difference from Gateway architecture, is that base station in this case does not perform significant data processing and aggregation.

– TCP/IP implementation in WSN, where TCP/IP stack is implemented in WSN network. Each node has an IP address and can send and receive packets from any host on the Internet.

The type of WSN architecture has a profound influence on security of communication between WSN sensor node and arbitrary host in the external network. For example, in case of Gateway Architecture achieving end-to-end security between host in external network and sensor node in the WSN is impossible, because base station, which acts as a gateway should be able to understand protocol WSN is talking and translate it to external networks protocols. Also, gateway should understand the transmitted data, to be capable of performing data aggregation before sending it to the host in external network.

In case of Overlay Architecture end-to-end security becomes possible, because WSN and external network are talking the same high-level, overlay protocol, while their transport protocols might be different. But of course in this case, base station cannot perform any data aggregation, because data might be encrypted end-to-end, in which case intermediaries are unable to understand it and hence processing is impossible.

In case of TCP/IP in WSN, achieving end-to-end security between WSN and external network is possible, because both networks operate the same transport level protocol. In this case, it is also reasonable to use standard transport level security protocols (such as TLS and SSH) for communication between WSN and external network. Although implementations of TCP/IP for sensor nodes already exist [4, 5] (uIP, uIPv6) and existing research has shown that implementing asymmetric cryptographic primitives for sensor nodes, with their limited capabilities, is possible (Elliptic Curve Cryptography or ECC, which allows that, will be discussed in section 3). There are still no known implementations of standard transport level security protocols for WSNs available. Research in that direction could bring WSNs and it’s application area, Internet of Things, to the next level, enabling secure end-to-end communication between standard TCP/IP hosts on the Internet and sensor nodes in WSNs.

From the discussion above, we can clearly see, that problem of security in WSNs can be divided into two sub problems. First is problem of secure communication/operation of sensor nodes inside WSN and second, secure communication of WSN and it’s sensor nodes with the external networks. The first problem was widely covered by researchers, multiple security solutions were proposed and there are existing implementations. Nevertheless, second problem, of secure communication between WSNs and external networks, was not yet widely covered by researchers.
2.1 Specific security requirements for WSN

WSNs have similar requirements to other networking systems [6, 7]. Confidentiality, Integrity, Availability and Authentication are nowadays required almost in every secure system. But in many WSNs, freshness, non-repudiation, detection of compromised nodes and physical security, are also strong requirements. Requirement of privacy/anonymity are of a smaller importance in WSNs. Further, we will discuss each requirement and its importance for WSNs separately.

- Confidentiality is of main importance in WSNs. As sensor nodes transmit data over wireless channels it is very easy for attacker to eavesdrop the data being transmitted. The main problem with end-to-end confidentiality in WSNs is it’s conflict with requirement of data aggregation, which is functional requirement in many WSNs. In order to aggregate data intermediate sensor node should be able to understand it. In case if data is encrypted, intermediate node can’t make any sense of data being transmitted. Hence currently the most popular encryption model in WSNs is hop-by-hop encryption. Nevertheless hop-by-hop encryption unveils the problem of compromised nodes. If attacker is able to introduce his own sensor nodes to the network or compromise existing nodes, he will be able to eavesdrop on all the ”encrypted” traffic transmitted through that sensor node. Another problem with confidentiality in WSNs is that setting up truly secure session key between two sensor nodes is not always trivial. In the Internet, asymmetric cryptography and Public-key infrastructure are usually used for setting up symmetric session key between entities. On sensor nodes, asymmetric cryptography might be not available and implementing secure key distribution in WSNs environment is a very challenging problem.

- Integrity in WSN is as important as confidentiality. Similarly to confidentiality, integrity is also implemented hop-by-hop in majority of WSNs.

- Availability is very critical in WSNs, especially in deployments which operate as alarm or intrusion detection systems. Consequences of Denial-of-service (DoS) attacks are much more serious than in modern IP networks. DoS attacks can drain batteries of some of the sensor nodes and thus block normal work of WSN. Also causing battery of one of the sensor nodes to drain, may result in unreachability of multiple sensor nodes. In this sense, base station is a serious bottleneck of WSNs, if attacker will be able to crash base station, whole WSN will be unreachable. In case of availability, hop-by-hop security is rather advantage, than disadvantage. On every hop, forwarded packet can be authenticated and in case of invalid authentication, dropped. This can stop DoS attacks from propagating into the network. Packet level authentication (PLA) [2] is one of the projects, which sets as a goal to create protocol where every packet is authenticated and can be validated at each hop in the network. As PLA is designed with WSNs in mind, it’s results can be also applied to WSNs.

- Authentication is another very important security requirement of WSNs. In order to provide confidentiality, integrity, availability and fraud node detection, strong authentication is necessary. State-of-the-art authentication
protocols in the Internet use asymmetric cryptography, for authentication purposes. Majority of WSNs use symmetric cryptography for authentication purposes, as asymmetric primitives are not yet widely available in WSN security frameworks. Certainly, only very powerful sensor nodes will be capable of performing computations required for asymmetric cryptographic primitives.

- Many WSNs operate as alarm systems, by triggering the alarm to work, in case if observable phenomena exceeds the threshold. Verification of freshness of a message might be especially important in that type of systems, to prevent false alarms by message replays.
- Non-repudiation can also be very important requirement in alarm system. For example, if sensor node has triggered an alarm to work, it should not be capable of disclaiming that action later.
- Detection of compromised nodes is also very important. As WSN is by it’s nature data oriented network. Sensor nodes, base station and operator of WSN should be capable of detecting, which of sensor nodes can they trust and which not. WSN should be capable of detecting compromised nodes and exclude their data from data aggregation process.
- WSNs are typically deployed in hostile environment, where attacker can have physical access to the sensor nodes. Hence physical security of sensor nodes is also of primary importance. In practice, it is impossible to prevent attacker from physically destroying a sensor node, but sensor nodes should be capable of identifying a physical attack and reporting to their neighbors and base station, that they cannot be considered trustworthy anymore. A lot can be learned in this area from Subscriber Identity Module (SIM) cards, which contain sensitive information, such as private keys and are physically located in ”hostile environment”. It is very difficult to get the sensitive information out of SIM cards (all sensitive computations are performed on SIM itself). And in case of small physical damage SIM card can easy block and become unusable. That type of designated computational chips, with sensitive information, can be also used in WSNs. If this module is damaged, sensor node can report to WSN that it is not trustworthy anymore. The only disadvantage of that kind of modules is significant increase in price of the sensor nodes

As we can see from our discussion about WSNs, WSN Security is all about trade-offs. We cannot have one property without sacrificing the other. Eventually, what security components you choose for your WSN deployment highly depends on a particular application and on security requirements of your system.

3 WSN Cryptography

Cryptographic primitives are cornerstone of every secure system. In this section, we will discuss how symmetric and asymmetric cryptographic primitives are used in modern WSN security protocols.
3.1 Symmetric

Majority of modern security solutions, in WSNs, are based on symmetric cryptography, the reason is much smaller usage of computational resources and energy efficiency. Until recently it was believed that asymmetric cryptographic primitives are unsuitable for WSNs, as they require too many resources.

Three major requirements, confidentiality, integrity and authentication, are addressed by majority of modern symmetric cryptography based solutions. In case of encryption, choice is usually between three block-cipher algorithms: Advanced Encryption Standard (AES), Skipjack and Rivest Cipher (RC5). In case of integrity and authentication Message Authentication Codes (MACs) in majority of popular systems are based on AES or RC5 ciphers in CBC-MAC modes.

The main problem with symmetric encryption, is secure session key distribution among sensor nodes. The most popular symmetric key based solution for that, in fixed networks, is Kerberos. Nevertheless Kerberos relies on session key distribution servers, which should always be online in order for system to function. Hence, Kerberos-like solutions are very difficult to deploy in WSN realities.

There are currently three most popular key distribution models defined for WSNs: network-wide key, group-key, pairwise shared key. They all are based on pre-distribution of symmetric master keys to the sensor nodes, before deployment. In case of network-wide key, all the sensor nodes share the same master key. Group key management allows, specifically defined clusters of sensor nodes, to use group-wide symmetric key for communication. Pairwise shared keys allow neighbour sensor nodes to exchange session keys with each other.

3.2 Asymmetric

Recent developments in Elliptic Curve Cryptography (ECC), have made usage of asymmetric key cryptography in WSNs viable. ECC is based on algebraic structure of elliptic curves over finite fields. The main advantage of ECC is providing similar security to RSA, with significantly smaller public/private keys and hence faster computation times. 160-bit key ECC provides similar level of security with 1024-bit long RSA key. In addition, analysis performed in multiple research papers has shown, that energy consumption of ECC based solutions is much smaller than of similar RSA based solutions [11, 12].

Availability of asymmetric cryptographic primitives, on sensor nodes, opens possibility for using hybrid encryption techniques in WSNs. Hybrid encryption means, that session keys between communicating entities are distributed using asymmetric cryptography and key exchange algorithms (e.g. Diffie-Hellman), and data itself is encrypted with symmetric session keys. The main advantages of hybrid encryption are secure authentication and key exchange using asymmetric cryptography and Diffie-Hellman key exchange, and usage of efficient symmetric algorithms for actual data encryption.

Some researchers have also proposed using endorsement-based mechanisms, to enable end-to-end security in WSNs and at the same time prevent DoS attacks from spreading into the network [12]. Endorsement means, that if two arbitrary
nodes, in WSN, want to communicate, they have to receive N endorsements from their neighbours before being able to send the packets. Technique described in [12] works based on Shamir’s secret sharing. Secret is divided into the shares and shares are predistributed among sensor nodes, at any point N shares of secret are needed to discover the whole secret. Technique works the following way, sensor node A which wants to communicate to remote sensor node B authenticates itself to N of his neighbours, using his private key, and acquires a share of a secret from each neighbour. Then it sums up the shares and gets secret S, which is used as a symmetric pairwise key with remote node B. Sensor node B also calculates secret S, by using his own share of S and access list it receives from sensor node A. The advantage of this scheme is that in order to figure out the secret key, used in WSN, attacker should compromise at least N nodes, instead of just one.

Currently it is difficult to evaluate what will be the future of asymmetric cryptography in WSNs, although there are existing implementation of ECC based asymmetric cryptoprimitives [9], they do not seem to be widely used in actual WSN deployments.

4 Existing Security Solutions

In this section we will evaluate three security frameworks for WSNs. TinySec, which is based on TinyOS operating system. ContikiSec which is based on Contiki operating system and ZigBee security protocol, which is independent from the underlying platform.

4.1 TinySec

TinySec is link-layer security layer for TinyOS operating system, the most popular operating system for WSNs. The reason why link-layer was chosen as a security layer is that in WSNs communication is usually one-to-many(sensor node to base station), which means that intermediate nodes need to aggregate and process data. TinySec is designed to be as lightweight as possible and has minimal impact on WSNs performance, implementing security layer which adds very little overhead to original TinyOS packet exchange. The main design principle of TinySec is to minimise the amount of security-related information sent over the network, in order to ensure energy efficiency of security system. It is much more energy efficient to perform additional computations on the sensor node, than sending extra bytes over the network.

TinySec addresses three main security requirements: confidentiality, integrity and authentication. Physical security of sensor nodes, replay attacks and key exchange protocols are not addressed by TinySec. TinySec support two modes of operation, TinySec-AE and TinySec-Auth. TinySec-AE provides confidentiality, integrity and authentication and TinySec-Auth provides only integrity and authentication. Fig.2 displays packet formats of TinySec and TinyOS. As we can see, additional overhead in TinySec-AE packet from TinyOS packet is 5
Fig. 2. Packet formats for TinySec-AE, TinySec-Auth and TinyOS default packet. Hatched fields are protected by MAC, gray parts are encrypted [8].

Bytes and additional overhead in TinySec-Auth packet is only 1 Byte. Analysis performed in [8] shows that TinySec-Auth adds 3% to total energy consumption when compared to plain TinyOS and TinySec-AE adds 10% to energy consumption. This increase in consumption is caused mostly by increased number of bytes transmitted, not by computations needed for cryptographic primitives. Confidentiality is achieved in TinySec using block-cipher encryption in Cipher Block Chaining - Ciphertext Stealing (CBC-CS) mode. TinySec implements three block-ciphers RC5, Skipjack and AES, by default Skipjack is used.

Important aspect of block-cipher encryption in CBC mode is choice of Initialization Vector (IV), which is needed to achieve semantic security of the messages (encrypting same message twice should give different ciphertexts). IV should be generated, such that it is as unrepeatable as possible. TinySec uses following IV structure dst || AM || l || src || ctr, where dst is destination address of the receiver (2 bytes), AM is message type (1 byte), l is payload length (1 byte), src is source address of the sender (2 bytes) and ctr is 2 byte counter. Thus IV is 8-byte value, which adds only 4-bytes to the original TinyOS packet, because dst || AM are already used in TinyOS packet and the only element added by TinySec is 4-bit src || ctr value. This makes IV values to repeat every 2^{16} times. Which in case if we send out one packet per minute makes IV value to repeat every 45 days. IV in TinySec is used in counter mode, which can cause problem when used with CBC mode, revealing some partial information about the plaintext. Authors of TinySec are using an easy fix for that problem, pre-encrypting IV before XOR-ing it with the first part of the plain text. Thus making IV in counter mode usable with CBC block-cipher mode and making created IV secure against birthday-paradox based attacks (in case of birthday-paradox attack, first repetition of the IV can be expected after 2^8 IVs generated).

Fig.3 shows encryption in CBC-CS mode used in TinySec. The main advantage of CBC-CS mode is that cipher text produced is the same length as
Fig. 3. Encryption in CBC-CS mode, with encrypted IV. Gray areas denote parts which will be included in cipher text.

the plain-text (in original CBC mode if plain-text is not equally divisible to blocks, on which block-cipher operates, then cipher text is longer than original plain-text). In figure we can also see that IV vector, which is 8-bytes is encrypted before usage. Block size in TinySec encryption is 8-bytes. Fig.4 shows decryption in CBC-CS mode for TinySec.

Integrity and Authentication is achieved in TinySec, using Message Authentication code in CBC mode (CBC-MAC). The length of a MAC used in TinySec is 4 bytes, which means that in order to forge the MAC, by bruteforce, attacker should make $2^{31}$ attempts to send the message which in real WSN environment would take very long time to perform and would rather drain batteries of a node and create DoS attack on WSN, than would allow to break the MAC of a message.

TinySec does not define any key exchange mechanism, network-wide symmetric key is just predeployed to every node, during code deployment. This makes TinySec vulnerable to attacks based on compromised nodes. Nevertheless authors claim, that more secure key exchange mechanism can be built on top of TinySec.

Building efficient key exchange mechanism for WSN is a big challenge. Current symmetric cryptography based solutions are communicationally too expensive, which by default make them unsuitable for energy-constrained sensor nodes and solutions based on asymmetric cryptography, although less communicationally intensive, create a lot of computational overhead.

Currently, TinySec is integrated into TinyOS. Original implementation of TinySec requires 728 bytes of RAM and 7146 bytes of program space, which makes it usable even with very constrained sensor nodes.
4.2 ContikiSec

ContikiSec is recently introduced, link-layer security solution for Contiki operating system. Design of ContikiSec is heavily influenced by TinySec, trying to be minimalistic and providing only basic security services. ContikiSec provides confidentiality, integrity and authentication. ContikiSec has three modes of operation: ContikiSec-Enc, ContikiSec-Auth and ContikiSec-AE. ContikiSec-Enc provides confidentiality only, ContikiSec-Auth provides integrity and authentication and ContikiSec-AE provides confidentiality, integrity and authentication.

Fig. 5 shows packet formats of ContikiOS, ContikiSec-Enc, ContikiSec-Auth and ContikiSec-AE. From figure we can see that ContikiSec-Enc adds only 2 Bytes overhead to basic ContikiOS packet, ContikiSec-Auth adds 4 Bytes and ContikiSec-AE adds 6 Bytes.

Similarly to TinySec, confidentiality is achieved using block-cipher in CBC-CS mode. Contiki uses AES instead of Skipjack, as it is proven to be more secure, although more computationally intensive. IV used with AES-CBC is 2-bytes long and is generated randomly. Which ideally means that repetitions should occur every $2^{16}$ times. But as IV is chosen randomly we can expect first repetition to occur, probabilistically after $2^8$ times, which means that IV generation might not be good enough for AES-CBC mode. Encryption/decryption schemes are similar to the ones described in figures 3 and 4, with only difference that IV is not encrypted before usage and it’s length is 2 instead of 8 bytes.

Integrity and authenticity in ContikiSec is very similar to TinySec and uses 4 Byte CMAC (which is variation of CBC-MAC). The reason for usage of 4 Byte

![Diagram of Decryption in CBC-CS mode, with encrypted IV.](image-url)
CMAC is that it would be impractical for attacker to online bruteforce 4 Byte CMAC in WSN environment, where bandwidths are small and battery life of sensor nodes is limited.

In case of ContikiSec-AE, where confidentiality, integrity and authenticity of messages are provided. ContikiSec uses special mode of block-cipher, Offset Codebook Mode (OCB), which is variation of authenticated encryption mode of operation. Advantage of OCB is that instead of performing two operations (Encryption, MAC) for achieving confidentiality, integrity and authenticity, only one operation (authenticated encryption) is performed, which reduces time of computation and achieves the same result. For example, to encrypt and authenticate 40 Byte payload, OCB needs about 20ms, while AES-CBC + CMAC need more than 25ms (on MBS-430 sensor platform).

Similarly to TinySec, ContikiSec does not define any key exchange mechanism, instead network-wide symmetric key is used, which makes ContikiSec vulnerable to attacks based on compromised sensor nodes.

4.3 ZigBee Security

ZigBee is a specification of communication protocols for low-power digital devices, which also include security related specifications. ZigBee covers wide range of devices, WSNs related specifications are only subset of ZigBee suite. ZigBee is not a particular implementation, but set of specifications to guide the implementation. Cryptographic primitives used with Zigbee are similar to the ones used in TinySec and ContikiSec, using AES-128 in CBC mode for confidentiality and CBC-MAC for integrity and authenticity. In addition to that ZigBee provides message freshness, by using counter [1].
The interesting part of Zigbee is its key agreement protocol. There are three kinds of keys in ZigBee's security specification: master keys, link keys, and network key. Master keys are pre-installed to each node and are used to exchange the link keys between nodes in the WSN. Link keys are unique between each pair of nodes, they are used to encrypt information between two communicating sensor nodes. Network key is a 128-bit key shared between all nodes in the WSN, it is updated by ZigBee Trust Center at different intervals. Network key is used for authenticating parties joining the WSN. Network key is updated after some periods, new network key is set up using old network key. Trust Center is the entity in ZigBee, which deals with key sharing and updating. Fig. 6 shows the usage of different keys in ZigBee WSN.

ZigBee trust center has support for two modes of operation: Residual Mode and Commercial Mode. Residual mode is more lightweight and thus is more often used with WSNs. In residual mode, trust center shares only network wide key with nodes in the network and data is exchanged between nodes using network wide key (approach similar to TinySec and ContikiSec, except that in latter ones network key is usually predeployed and rarely updated). In commercial mode, trust center shares also master and link key with devices in the network, this mode requires more storage on the nodes and thus is less preferable for WSNs.

5 Conclusions

In this paper we have reviewed the security issues in WSNs. Security in WSN is a difficult issue including not only security of transmitted data, but also physical security of sensor nodes. Eventually choice of security requirements is different for every particular application of WSN. Currently the most popular security
primitives used in WSNs are symmetric as well as most of existing security solutions are based on symmetric cryptography and fairly trivial symmetric-key predistribution approach. We have reviewed three of most popular existing implementations/specifications for WSNs: TinySec, ContikiSec and ZigBee security. Existing implementations of security primitives for WSNs have shown, that designing security for WSNs is interesting engineering problem, where specifics of WSNs should be considered. Specifics of WSN are not only disadvantageous for security, but also can be advantageous, for example 4-byte MACs are usually suitable for WSNs, while in fixed-networks they would be considered extremely weak.

A lot of attention, in research community, is also paid to the usage of asymmetric cryptography in WSNs, because of it’s inherent suitability for WSNs. Nevertheless there are not many usable implementations, of asymmetric cryptography primitives, for WSNs.

References
