Cross Layer Security Frameworks for Wireless Sensor Networks

Seeger Jan
Betreuerin: Schmitt, Corinna
Seminar Innovative Internet Technologien und Mobilkommunikation WS09/10
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: seeger@in.tum.de

ABSTRACT

A Wireless Sensor Network (WSN) is a tool with many applications. Because of its characteristic structure and hardware composition, it is much more difficult to ensure authentication, integrity and confidentiality in WSNs. Several algorithms have been proposed to fulfill these requirements. However, securing each OSI layer individually leads to inefficiencies in the operation of the network and security measures not suitable for the diverse environments WSNs are deployed in. The two cross-layer frameworks UbiSec&Sens and the ISA framework compared in this paper aim to reduce duplication, increase energy efficiency and provide flexible security primitives adjustable for a wide array of requirements. Both frameworks are usable in different environments and increase the flexibility, energy efficiency and security strength of a network in contrast to a layered security approach.

Keywords

Wireless Sensor Networks, Cross-Layer Security, Security Frameworks, Ubisec&Sens, Integrated Security Architecture

1. INTRODUCTION

A Wireless Sensor Network consists of small, general-purpose, low-power computing nodes connected over a wireless link. It is used for sensing and aggregating environmental data in areas such as agriculture, the military and environmental monitoring. One example application is prevention of forest fires by monitoring data such as humidity and temperatures.

A sensor node consists of a low-power general processor (such as an Atmega 128L), a wireless transceiver (for example using the IEEE 802.15.4/Zigbee wireless stack), different sensors and a battery. Processing power is very low and memory is scarce. To illustrate: A modern personal computer has about 400 times as much processing power and more than 5000 times more available memory space. Energy is also a concern: The energy ratio between sending, computing and sleeping is about 170:8:1 in a node. It is therefore sensible to minimize sending and computing operations.

These tiny sensor nodes are organized in a tree- or star-like fashion: Each leaf node is linked to an aggregator node, which in turn can be linked to yet another aggregator node. The last layer of aggregator nodes is then linked to the base station, as illustrated in Figure 1. This structure leads to a traffic flow from the leaves over the aggregator nodes to the base station. Only a negligible amount of data passes back from the base station towards the nodes, compared to the amount of data flowing in the reverse direction.

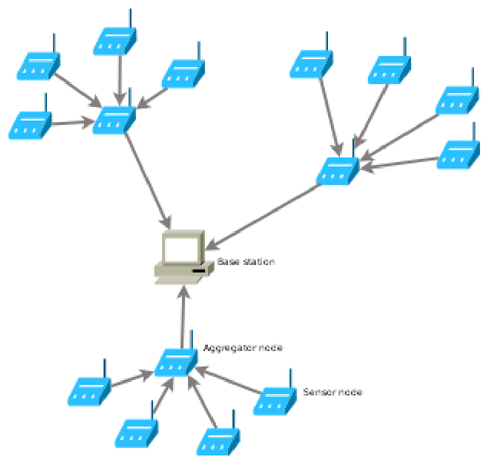

Figure 1: Structure of a wireless sensor network

In order to protect the potentially sensitive data being passed through the network, strong security measures need to be taken. Security in wireless sensor networks is an active research topic, and many algorithms have been proposed for use in sensor networks.

Because of the very weak hardware, security algorithms need to be as computationally lightweight as possible, something that is not given in most "standard" algorithms and cryptographic protocols such as Kerberos or TLS.

Also, the non-tamper-resistance of the single nodes poses additional challenges: Any secret material contained in a single node can be gained by an attacker by simply stealing the node and reading its secret key store. The Dolevt-Yao threat model of an attacker situated *between* two personal computers is thus not applicable in wireless sensor networks [4].

One solution for these challenges is to integrate the different layers of the traditional security architecture. This approach is called *Cross-Layer Security.* Cross-Layer Security aims to provide integrated security services to its host by making security decisions across all layers.

Two Cross-Layer Security frameworks have been selected for this overview: The European Union "UbiSec&Sens" [2] framework, and the "ISA"-Framework envisioned by members of the Sikkim Manipal Institute of Technology [9].

2. THE UBISEC&SENS FRAMEWORK

The UbiSec&Sens project (full name is "Ubiquitous Security and Sensing in the European Homeland") is a research project supported by the European Union as a "Specific Targeted Research Project" with such partners as NEC and the RWTH Aachen [1]

It targets medium- to large-sized Sensor Networks and aims to provide an integrated and customizable security environment for sensor nodes. In the following sections, one UbiSec&Sens configuration will be explored and relevant algorithms will be explained.

2.1 Authentication

Authentication is tricky in a WSN: Any unique knowledge by which a node can prove that it is, in fact, itself and not an attacker, can easily be stolen by an attacker: Because the nodes are not tamper resistant, the attacker can simply open the node, and extract the key from its storage location. Asymmetric cryptography is no help here, since the private key can compromised just as easily.

One possible idea is reducing the strength of authentication: Instead of identifying an arbitrary node, we merely try to *re-identify* a node that has provided a certain service. This property is called "Zero Knowledge Authentication" (ZCK) by Weimerskirch and Westhoff [7].

Their proposed algorithm works as follows:

1. First, nodes A and B generate a public key from their global private keys, the identity of the other node and some random data.
2. Then, the hash function H is applied n times to the global private key to create the public keys, x_n^A for A and x_n^B for B.
3. The first time A and B communicate, their public keys x_n are exchanged.
4. The next steps are repeated for every communication between A and B.
5. A sends x_{n-1}^A to B, B checks if $H(x_{n-1}^A) = x_n^A$.
6. B sends x_{n-1}^B to A, A checks if $H(x_{n-1}^B) = x_n^B$
7. Finally, each node stores the last key that was sent to them, so instead of storing X_n^B, A stores X_{n-1}^B after this authentication process.

[1]See `www.ist-ubisecsens.org`

ZCK authentication is claimed (without proof) to be the strongest possible authentication in a WSN by Weimerskirch and Westhoff [7].

The attacker can still impersonate a node by reading its key store, but if he does not provide the service connected to the key, contact with that node will be broken. False data injected by the attacker needs to be filtered out by the aggregation algorithm.

The use of one-way hash functions allows a comparatively high system security coupled with low computing requirements compared to authentication with preshared keys as practiced by the ISA framework (see chapter 3.2).

The ZCK authentication algorithm thus provides for a computationally lightweight authentication solution for wireless sensor networks which nevertheless fulfills the needs of other higher level algorithms.

2.2 Encryption

There are several encryption algorithms that are considered acceptable for running on a node: As seen in chapter 3.3, "standard algorithms" such as RC5 can be run on a sensor node.

However, using a standard encryption algorithm such as RC5 or even an asymmetric cryptosystem like ElGamal disregards the main purpose of a Wireless Sensor Network: It is the aggregation and collection of data. Most of the energy and processing time expended in a Wireless Sensor Network is dedicated to either sensing data or performing operations like median, average or movement detection on that data.

For such cases, using a *Privacy Homomorphism* (from now on abbreviated "PH") grants large advantages: A PH is a cryptosystem with encryption function $E_k(x)$ and decryption function $D_k(x)$, and an operator $\oplus$ such that the following condition holds:

$$D_k(E_k(x \oplus y)) = D_k(E_k(x) \oplus E_k(y)) = x \oplus y$$

If the $\oplus$ operator is $+$, the homomorphism is called "additive homomorphism", if $\oplus$ is $*$, the homomorphism is called "multiplicative homomorphism". This property allows the aggregation of data without decryption of the received packets. This is called "Concealed Data Aggregation" (CDA) in [8]. CDA saves energy in the nodes, because packets do not have to be decrypted in order to be processed. Also, concealed data aggregation protects the transferred data in the case of an attacker capturing an aggregator node.

The following additive and multiplicative PH proposed by Westhoff et al. in [8] is part of the UbiSec&Sens modules.

The symmetric PH proposed by Westhoff et al. is a probabilistic homomorphism: First, choose the public parameters d and g, where d is larger than 2, and g has many small divisors and integers with an inverse element smaller than g. The secret key is then the pair $k = (r, g')$ where r has an inverse element in $\mathbf{Z}_g$, and $g = g'^n$ where n is an integer. Then, partition the sensed value S into random parts a_i, so that $\sum_{i=0}^{d} a_i \mod g' = S$. Then, the value is encrypted by calculating $E_k(a) = (a_1 r \mod g, a_2 r^2 \mod g, \ldots a_d r^d \mod g)$. To decrypt, simply use the inverse element r^{-1} to compute each a_i and then sum up these values: $D_k(a) = \sum_{j=1}^{d} a_j * r^{-j} \mod g'$. The encryption is akin to transforming the sensed value into an number system with an unknown base.

This algorithm is additively homomorphic: Simply sum up the encrypted parts to gain the new encrypted value: $a+b = c_i = a_i + b_i \mod g$. To multiply, proceed similar to manual multiplication: $a * b = \forall i \neq j : c_{i+j} = a_i * b_j \mod g$. Then simply add up the parts with the same index.

However, since this is a symmetric algorithm, a special key distribution algorithm needs to be used in order to prevent an attacker from gaining a key and decrypting the data flowing through the network too easily.

2.3 Key distribution

Connected to chapter 2.2 is the key distribution: Because every sensing node needs a key to encrypt its data, keys need to be distributed prior to starting aggregation.

Asymmetric cryptography would be optimal. However, asymmetric cryptography poses hardware requirements that a sensor node is not able to fulfill, especially not for encrypting sensed data (which is a very frequent operation).

A pool of preshared keys would also be feasible. But these keys would have to be configured before deployment, and their distribution most probably would not mirror the geographic distribution of the nodes.

“Topology aware group keying”, proposed in the same paper as CDA [8], solves this problem: It works with a pool of preshared keys on each node, creates a key distribution coincident with the geographic distribution of the nodes and retains as little sensitive data as possible on each node.

It works in the following way:

1. Before deployment, each node is configured with the same preshared key pool. Each key in that pool has a unique key ID.
2. When deployed, the nodes with distance 1 to the base station start broadcasting a list of key IDs randomly selected from the key pool and deletes its whole key pool.
3. Any node that receives an ID broadcast either deletes its whole key pool or randomly chooses a key from the broadcast ID list.
4. It rebroadcasts the ID list, and ignores all subsequently received key lists.
5. Finally, all nodes that have not received a broadcast delete their whole key pool.

Each node n with distance 1 from the base station create a “routeable region”: In this region, there is a certain number

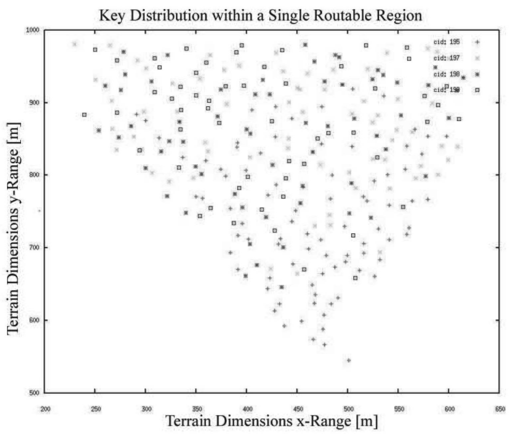

Figure 2: Key distribution in a single routeable region [8]

of nodes that have the same keys. Then, the nodes are operated in a time-slice fashion: Each time slice, only the set of nodes that have a certain key operates as sensing nodes. The rest can either work as aggregator nodes or save energy by going into standby mode. For a visualization of the key distribution, see Figure 2: Each different mark represents a different chosen key. The density of the key distribution can be controlled in step 3: By varying the probability with which the node deletes its key store, the private information stored in the nodes and the aggregator node density can be controlled.

It is clear, that such a distribution of the keys grants many benefits: An attacker needs to capture many nodes before he can decrypt the traffic in every time period. Since the probability of capturing a useful key decreases with the number of keys already obtained, even with a high key density, the attacker must expand a comparatively high amount of time to gain full access to the encrypted data.

The average number of nodes needed is based on a modified coupon collector’s problem. The probability to find a new key decreases with the number of keys already found. Since there is also a number of nodes that do not have any keys in their key stores, it can be said that the average number of nodes to collect for full network penetration for n nodes is higher than the nth harmonic number H_n.

2.4 Data storage

Most sensor networks do not require storage of information: Sensed data is immediately passed to the aggregator nodes, the aggregation function is calculated (perhaps several times in different levels of the sensor network) and the result is passed on to the base station.

However, in certain kinds of sensor networks, data storage is useful: In impassable regions, connectivity to researchers may not be given, and the data needs to be stored until someone retrieves the data. Also, redundant data storage increases the resilience of a network towards loss of sensor nodes: In a non-storing sensor network, once a significant portion of sensors is destroyed, the whole network becomes unusable. With distributed storage inside the sensor nodes, the data can still be retrieved.

"Tiny Persistent Encrypted Data Storage" (from now on called TinyPEDS) described by Westhoff, Girao et al. [5] works by partitioning the sensor network into quarters and then distributing the data in these quarters to achieve redundancy.

Notable is the use of two PH's: One symmetric PH for immediate encryption and transfer (here, the PH described in chapter 2.2 can be used), and an asymmetric PH for long-term storage encryption.

The algorithm works as follows: During the startup phase, the network is divided into quarters: Each aggregator node n_i has "next node" n_{i+1}, which redundantly stores its data d_i. When sensing, the aggregator node sends the symmetrically encrypted data to its next node, where it is added to the nodes' own encrypted sensing value. This value is then encrypted and stored using the asymmetric homomorphism. So node n_i stores $E_s(d_i), E_a(d_i + E_s(d_{i-1}))$, where E_s is the symmetric homomorphism, and E_a is the asymmetric homomorphism.

During a normal query, the query packet is flooded to all nodes. Each node tests if it is included in the node set contained in the query, and if it is, sends the requested encrypted data $E_s(d_i)$ back to its aggregator node. The answer is then aggregated and percolates upwards through the tree to the base station.

If the sent query is not answered in a certain time period, it is presumed that a part of the sensor node is lost, and a *disaster query* is run. If data from node n_i is requested, each node n_j tests if it is *not* included in the requested node set. It then sends back its redundant data $E_a(d_j + E_s(d_{j-1})), E_s(d_j)$. The base station then receives three values, from which it can recompute the requested value.

TinyPEDS thus allows resilient and energy efficient data storage in a wireless sensor network.

3. THE ISA FRAMEWORK

The ISA framework from [9] pursues a different concept from UbiSec&Sens: Instead of creating an open toolbox of algorithms that are designed to fit together, the ISA framework takes a "one-size-fits-all" approach.

"ISA" stands for *Intelligent Security Agent*, an added component in the node graph (see Figure 3) that is responsible for all security decisions. It is comparable to a TPM in a personal computer and allows integrated operation of all security services.

3.1 Tasks of the ISA

The ISA is the sole controller of all security options. Whenever a packet is sent or a communication channel is opened, the ISA is consulted in regards to security. Also, the ISA fulfills several other tasks necessary for secure operation of the network.

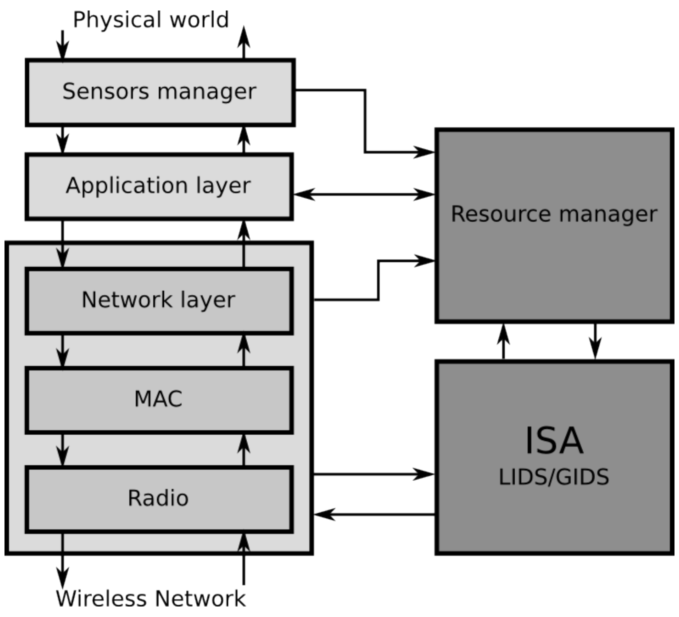

Figure 3: The node component graph with ISA added

Since ISA aims for component-based security (as opposed to UbiSec&Sens, which takes a probabilistic approach), each nodes' ISA monitors several important parameters that help to detect local and global intrusion attempts. These roles are called LIDS and GIDS respectively, which stand for "Local" or "Global intrusion detection system".

Among the monitored parameters for the LIDS are the sensed values, the packet collision ratio, the invalid packets and the number of retransmitted packets. If any of these parameters changes abruptly, the ISA reports to the base station.

The GIDS subsystem monitors parameters such as power consumption, signal strength of neighboring nodes and IDs of neighboring nodes, and also reports any anomalies to the base station.

Apart from intrusion detection, the ISA also stores a "security level" or "security precept": Whenever an abnormality is detected, the security level is changed to answer the perceived threat. This allows the network to provide only necessary security: When there are no attackers, expensive encryption operations do not make sense.

3.2 Authentication

Authentication in an ISA sensor network is done with traditional preshared keys: Each node has a preshared key together with each other node. This leads to higher space requirements. But the process of a challenge-response authentication as described below is well understood and researched, whereas the ZCK algorithm from chapter 2.1 is a comparatively new algorithm.

No special authentication algorithm is given in [9], but a simple challenge-response protocol like the following is easily implemented:

1. Node A wants to communicate with B. It sends a random number r_A to B, authenticated and encrypted with the preshared key $K_{A,B}$.

2. B receives that number and increases it by 1. Then, it generates another random number r_B. Both numbers are encrypted and authenticated and sent back to A.

3. A receives both numbers. It checks if B sent back the correct number $r_A + 1$ and sends $r_B + 1$ back to B.

4. B again checks the received number.

If any of the checks fail, the communication channel is torn down.

This challenge-response protocol is supported by a reputation based approach taken from [1]: The ISA saves all the past communication with a node and decides whether to cooperate with it based on several metrics: The signal strength of a node, the number of generated versus the number of forwarded packets and the number of dropped packets.

This allows a flexible infrastructure with changing sensor distributions. Also, an adequate ruleset leads to non-cooperative nodes to be disconnected from the network.

3.3 Encryption

The encryption proposed by [9] is selected by the ISA: Based on the "security precept" (see chapter 3.1) of the ISA, different security algorithms and parameters are selected.

In the paper, two algorithms are proposed: A simple XOR encryption which is not described in more detail, and the RC5 algorithm.

The XOR encryption can be implemented with a simple feedback shift register or a simple PRNG, where the data is XORed with the key stream. Although the cryptographic strength of that encryption is very low, it is, however energy saving and computationally lightweight.

RC5 is an algorithm proposed by R. L. Rivest in [6] which is computationally lightweight and highly configurable. It's basic structure is that of a regular feistel chiffre. A novel feature is the use of "data-dependent rotation": One half of the ciphertext is rotated by the other half of the ciphertext, as can be seen in algorithm 1. RC5 has been studied exhaustively, and is generally thought to be secure by current standards.

Because of it's simplicity, the source code can be included here (see algorithm 1).

```
A = A + S[0];
B = B + S[1];
for $i = 1$ to $r$ do
    A = ((A $\oplus$ B) $\lll$ B) + S[2 * i];
    B = ((B $\oplus$ A) $\lll$ A) + S[2 * i + 1];
end
```

Algorithm 1: The RC5 algorithm

Derivation of the key material S from the original key is a bit more complex. However, expansion of the key material needs to be done only once for each key.

RC5 is highly configurable, and the ISA uses that property to configure different security settings based on the perceived threat level. Security configuration is highly configurable, but in the ISA paper, four levels are proposed:

Level 0	Simple XOR "encryption"
Level 1	RC5 with 80 bit key and 4 rounds
Level 2	RC5 with 80 bit key and 8 rounds
Level 3	RC5 with 80 bit key and 12 rounds

Note that the key size is not changed. This is because increasing the key size would either mean redistributing keys, or distributing larger keys in advance. Both is a highly energy inefficient operation and so the key size is not changed.

3.4 Key distribution

The use of preshared keys requires an intelligent key distribution algorithm. Since the number of keys increases quadratically with the number of nodes, it is helpful to generate the node-to-node keys from a central master key kept in the base station. For that, a procedure taken from [3] is used: It allows the computation of pairwise keys from a central master key. However, it does not easily allow derivation of the master key from the pairwise node keys unless a certain number of users cooperate, i.e. until a certain number of nodes have been captured.

The approach is taken from coding theory:

Let n be the number of nodes in the network. Then create a public $k \times n$ encoding matrix G that is an MDS code, i.e. that has the maximum possible hamming distance between code words. The base station creates a private key matrix D. Then, the global key matrix K is given by $K = (DG)^T G$. If user i wants to communicate with user k, the column $K_{i,j}$ is chosen. Since K is symmetric and G is publicly known, it suffices to send $(DG)_i^T$ to node i in order to allow it to communicate with any other node. Because G is an MDS code, it is not possible to compute D from less than k keys.

On the other hand, once the attacker has captured k nodes, he can deduce D and so has access to all pairwise keys between all nodes. This contrasts with the UbiSec&Sens approach in chapter 2.3, where there is only a certain probability (which rises with the number of captured nodes) of network compromise.

4. CONCLUSION

ISA and UbiSec&Sens aim to provide the same benefits to a wireless sensor network. However, from the preceding chapters, it should be clear that both frameworks use different techniques to reach their goal.

UbiSec&Sens aims to provide an "economic" security level, i.e. provide the necessary security with as little overhead as possible. This is achieved by choosing from a carefully selected pool of algorithms which are tuned to work together as well as possible. This allows high energy efficiency and a long lifetime of the sensor network. However, it must be remembered that, at least in the described implementation, UbiSec&Sens does not aim to provide “hard” security, i.e. security comparable to a personal computer.

ISA on the other hand tries to ensure component based security. Each node monitors itself for possible intrusions and alarms the base station. While achieving component based security is a difficult target in a WSN, the precautions of the ISA provide more secure components than the described UbiSec&Sens implementations, which does not concern itself with compromised nodes in a local manner. Also, the ISA security precautions are highly flexible: The security level of the total network changes depending on the needed security level. This allows easy deployment and low configuration effort. Note however that the addition of a new component in the sensor node makes it impossible to use standard nodes for an ISA network.

ISA and UbiSec&Sens are both able to provide energy efficient security to a wireless sensor networks. However, both frameworks are applicable to different environments: UbiSec’s modularity at deployment time and high energy efficiency makes it suited to long-term deployment, while ISA is better used in large networks that are highly dynamic, both in their security requirements and network structure.

5. REFERENCES

[1] A. Agah, S. Das, and K. Basu. A game theory based approach for security in wireless sensor networks. In *2004 IEEE International Conference on Performance, Computing, and Communications*, pages 259–263, 2004.

[2] F. Armknecht, A. Hessler, J. Girao, A. Sarma, and D. Westhoff. Security Solutions for Wireless Sensor Networks. In *17th Wireless World research Forum meeting*. Citeseer, 2006.

[3] R. Blom. An optimal class of symmetric key generation systems. In *Proc. of the EUROCRYPT*, volume 84, pages 335–338, 1985.

[4] D. Dolevt and A. Yao. On the Security of Public Key Protocols·.

[5] J. Girao, D. Westhoff, E. Mykletun, and T. Araki. TinyPEDS: Tiny persistent encrypted data storage in asynchronous wireless sensor networks. *Ad Hoc Networks*, 5(7):1073–1089, 2007.

[6] R. Rivest. The RC5 encryption algorithm. *Dr Dobb’s Journal-Software Tools for the Professional Programmer*, 20(1):146–149, 1995.

[7] A. Weimerskirch and D. Westhoff. Zero common-knowledge authentication for pervasive networks. *Lecture Notes in Computer Science*, pages 73–87, 2004.

[8] D. Westhoff, J. Girao, and M. Acharya. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution and routing adaption. *IEEE Transactions on mobile computing*, October 2006.

[9] K. Yadav, K. Sharma, and M. K. Ghose. Wireless sensor networks security: A new approach., 2008.