Secure Communications for Event-Driven Wireless Sensor Networks

Norman Dziengel, Nicolai Schmittberger, Jochen Schiller, Mesut Günes
Department of Mathematics and Computer Science
Freie Universität Berlin
Takustr. 9, 14195 Berlin, Germany
{dziengel, schmittberger, schiller, guenes}@inf.fu-berlin.de

Abstract

Existing security systems for Wireless Sensor Networks are either not able to cover all security requirements or are seriously affected by too high communications expenses to enable realistic deployments. We present a new security system for event-driven Wireless Sensor Networks, called PaRSec, that covers all security requirements with adequate expenses, for the first time. For this purpose, a definition of security based on the state of the art is given and the conformity and performance of the existing systems is analyzed. We introduce a concept that solves the typical problems of the existing systems – namely key management, completeness and expenses.

As part of the AVS-Extrem project for distributed event-detection in Wireless Sensor Networks, PaRSec's performance is benchmarked in a lab- and a field-test. The evaluation of the latency, throughput and energy consumption proves that in spite of the complete coverage of all security requirements PaRSec's expenses are still low enough to warrant a realistic deployment of an event-driven Wireless Sensor Network.

Keywords: WSNs, security, key management, distributed event detection, energy constraints

1 INTRODUCTION

With the advent of real-world applications in the area of Wireless Sensor Networks (WSNs) [1], the need for applicable secure communications increasingly moves into the attention of research. In wired networks, there are well established techniques to detect an intrusion and the cables additionally offer a spatial limitation to the accessibility. Wireless communications can be accessed directly within the radio range of any access point of the network by anyone. Catastrophe early warning systems or area-surveillance systems are two of the manifold examples of WSN applications that help to protect valuable goods as well as human life. As the operation of these systems must not be compromised by the intentional or accidental attacks on the system, a reliable security system is of high importance. The operation of an event detecting WSN is jeopardized by many factors. For one part, people may - intentionally or not - physically destroy one or more sensor nodes and thereby, e.g., cause the loss of data. However, the most important factor is the wireless medium which is easily accessible by anyone and thus does not offer any protection against malicious adversaries. Once connected to the network, the adversary could simply do an eavesdropping or passive traffic analysis attack which would not be noticeable or he could actively manipulate the network by removing, inserting or modifying packets in the network. Considering this, it is quite obvious that the sensor nodes need a security system to mitigate these threats in real-world systems.

This paper contributes to the field of security systems for WSNs in the following ways:

- A new and innovative key management scheme with a hybrid concept of pairwise- and global-keys is introduced.
- A new and real-world deployable layer 2.5 security system, PaRSec, based solely on symmetric cipher-algorithms and covering all security requirements is presented.
- One lab- and one field-test prove the applicability of PaRSec in real-world environments which has not been achieved before.

2 SECURITY REQUIREMENTS

There are certain security requirements defined by the state of the art that a security system needs to cover in order to be able to call itself secure. In [3], [4], [7], [10] they are defined as:

- **Confidentiality:** Only authorized sensor nodes may have access to the protected data.
- **Authenticity:** The identity of all participating entities, especially of the senders, must be known.
- **Integrity:** The data must not be manipulated or the manipulation must be recognized.
- **Freshness:** Packets must not be replayed or the replay must be recognized.
- **Semantic Security:** Multiple encryptions of the same plain text must not allow any inference about the plain text.
- **Availability:** The network must always be accessible, even in the case of an attack or similar.
- **Access Control:** Only authorized sensor nodes / persons may have access to the network itself.

		SPINS	SSNfPP	TinySec	PaRSec
Requirements	➢Confidentiality	✓	✓	✓	✓
	➢Authenticity	✓	✓	✓	✓
	➢Integrity	✓	-	✓	✓
	➢Freshness	✓	✓	-	✓
	➢Semantic Security	✓	✓	✓	✓
	➢Key-Management	✓	✓	-	✓
Expenses	➢Communication	⬆	⬇	➡	⬈
	➢Storage-Needs	⬆	➡	➡	➡
	➢Synchronization	⬆	⬆	⬇	⬇
Realized	➢Concept	✓	✓	✓	✓
	➢Simulated	✓(part.)	✓	✓	✓
	➢Implemented	-	-	✓	✓

Figure 1: Comparison of the related work with PaRSec

Since these are universal security requirements, they apply to WSNs just as they apply to any other area. But the wireless medium gives us some limitations on the feasibility of realizing them in WSNs. While confidentiality, integrity, authenticity, freshness and semantic security can generally be realized, availability and access control are somewhat different. As the wireless medium is a shared medium, accessible by anyone in range, access control on the physical level can simply not be done. On the other hand, one could also state that the access control is already covered by the authenticity and confidentiality requirements. With availability it is more a question of how much effort and expenses one is willing to spend. While, for example, a radio jamming device can render a typical sensor network dysfunctional, there are techniques like ultra-wideband (UWB) that can still operate even in those conditions but need very complex and expensive hardware.

Since the physical layer is not in the focus of this paper though, we will exclude availability and access control from further considerations and leave it up for discussion.

3 RELATED WORK

The need for a security system for WSNs has been recognized and worked on for quite some time. As it is a non-trivial task, we will see that each approach finally suffered from either unrealistic high expenses or an incomplete coverage of the security requirements leading to serious security issues. We distinguish expenses in communication, storage-needs and synchronization.

SPINS [10] is a protocol family designed to be universally applicable. It consists of three individual protocols, namely SNEP, Counter Exchange Protocol and µTESLA. The authors define three communication profiles that SPINS can secure:

- Sensor node → Base station (unicast)
- Base station → Sensor node (unicast)
- Base station → Sensor node (broadcast)

The SNEP protocol is used to secure the two unicast communication profiles. It uses a symmetric cipher-algorithm in Counter Mode (CTR) as well as a counter for freshness and a Message Authentication Code (MAC) for integrity and authenticity. The counter is also used to provide a loose replay protection. Instead of transmitting it with every packet, the SNEP protocol keeps the state of the counter on every sensor node. Although this reduces the additional overhead on each packet, it introduces the need for synchronization which is taken care of by the Counter Exchange Protocol.

The µTESLA protocol is used to secure the third communication profile listed above, i.e. it is used to provide authenticated broadcast communication from the base station to the sensor nodes. To be able to do this, several prerequisites need to be fulfilled. Firstly, all sensor nodes need to be time-synchronized. Secondly, all sensor nodes need to have a common master-key and, thirdly, they also need to know the key disclosure schedule for the one-way key chain generated by the base station.

In summary, SPINS does actually cover all security requirements, but the needed time- and counter-synchronization as well as the need to keep state of the counter and a missing secure broadcast communication for the sensor nodes make a realistic deployment doubtful. As there is neither an implementation nor a complete simulation this stays out in the open.

SSNfPP [2] is the abbreviation for "Secure Sensor Networks for Perimeter Protection" which is a system specially designed for the surveillance of close-by surroundings. Through its strong specialization to this particular area, it achieves a very lightweight system with an attack detection, source-routing and load-balancing. Each feature comes at the price of reduced universality or higher expenses though. For example, a more or less completely static network and time-synchronization are required for the system to work. Additionally there is no MAC or any equivalent mechanism to provide for the integrity of the packet and thus there is no complete coverage of the security requirements (see next section). However, an interesting concept introduced here is the combination of pairwise- and global-keys which allows broadcast traffic on the one hand and still keeps the possibility to communicate over the more secure pairwise-keys on the other hand. We will discuss this in more detail in section 5.2.

TinySec [7] was specially designed for the TinyOS operating system [6] and is thus also intended to be universally applicable. The authors laid their focus on a low complexity and a high flexibility of the system. They also introduced two different security-levels, TinySec-Auth and TinySec-AE, for authentication only and authentication and encryption respectively. This enables a convenient choice of the desired trade-off between security and extra expenses. Similar security-levels are also integrated in PaRSec but in regard of the available space, we will not discuss them here. The overall architecture of TinySec is, as

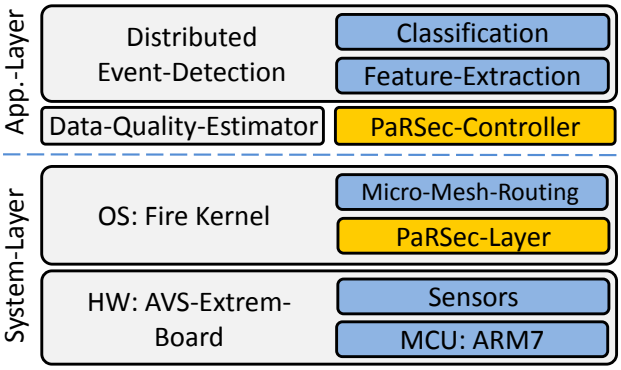

Figure 2: Architecture of AVS-Extrem with PaRSec

desired, very lightweight and requires no synchronization of any kind which reduces the additional expenses of the system significantly. Unfortunately, the authors decided to prefer the low complexity over completeness and deliberately left the freshness requirement uncovered. In addition, the key management, which is one of the main features of a security system (see section 5.2), was also left completely open. Although these points denote some serious issues of the security system, it is the only considered system that is, firstly, fully implemented and, secondly, completely open source. This makes it a good basis for own extensions and/or improvements.

Figure 1 summarizes the achievements and shortcomings of each system presented in the previous paragraphs again. As can be seen, only PaRSec achieves full coverage with adequate expenses. In addition, it is fully implemented and also open source.

4 DISTRIBUTED EVENT DETECTION

Distributed event detection [13] means to recognize environmental events by the help of multiple and collaboratively working sensor nodes. An event may be everything from an earthquake or an intruder climbing over a fence, up to a malfunction in a factory that needs to be detected. As we are researching in the area of WSNs, the challenge is to solve this problem by the usage of a minimum amount of radio communications and in-network evaluation. In general, this can be solved by local data processing as described in [1]. Typical approaches of event detection in WSNs collect raw data and then send this data to a base station to evaluate occurring events. In contrast to this, we apply a complete pattern recognition system on each involved node that is able to extract characteristic features of the upcoming data stream during an event. An integrated and patented feature selection algorithm [14] decides which features need to be extracted a priori. The features are distributed to the local neighborhood via broadcast and fused to a combined feature vector. This fused feature vector is then evaluated with the Euclidean-distance-based Nearest Prototype Classifier [13]. If the Euclidean-distance of the feature vector indicates that the unknown event can be mapped to one of the trained prototype vectors, it is recognized; if not, it is ignored. The next step filters whether the recognized event is of interest for the scenario or not. Only if the event is of interest, it will be sent via multi-hop communication through the WSN to an appropriate base station to allow the application or person in charge to decide whether to raise an alarm or not. The preceding procedure clarifies that communication is reduced to three packet types that we need to support. Firstly, the unavoidable communication for maintenance has to be taken into account. Second, the in-network communication during an event distributes a compressed feature vector. And finally, a possible but not mandatory event that needs to be send over multi-hop routes. Hence, our distributed event detection system which is applied in the AVS-Extrem fence surveillance project [5] reduces the amount of communication to a minimum. Due to this we do not have massive data-streams that have to be en- and decrypted.

5 PARSEC

In this section we will introduce the security system developed for the AVS-Extrem distributed event detection system - PaRSec. The focus of the system lays on a complete coverage of the security requirements while keeping the need for a real-world deployable system in mind. As we have seen in section 3, this is not an easy task and, to the best of our knowledge, it has not been achieved so far.

5.1 Fulfilled Security Requirements

As depicted in Figure 3 (left), PaRSec achieves complete coverage of all security requirements without introducing high expenses by applying the following techniques to each network packet:

Confidentiality is achieved by encrypting the payload part with a symmetric cipher-algorithm in CBC-Mode [9].

Authenticity and **Integrity** are achieved by calculating a message authentication code (MAC) over the entire packet with the already encrypted payload. This MAC is then appended to the end of the packet.

Freshness is achieved by introducing a counter into the header of each packet. By comparing this counter to previously received counter-values a loose replay protection is realized.

Semantic Security is achieved by inferring an initialization vector (IV) from the header of the packet and using this IV for the encryption of the payload as well as for the MAC. Hence, as long as the header does not repeat, which is the case as long as the counter does not repeat the ciphertext will always be different and thus suffice the rules of semantic security.

5.2 Key Management

One of the most essential components of a security system is the key management component. Even a perfect security system is of no use if the keys for it become publicly available or can not be efficiently

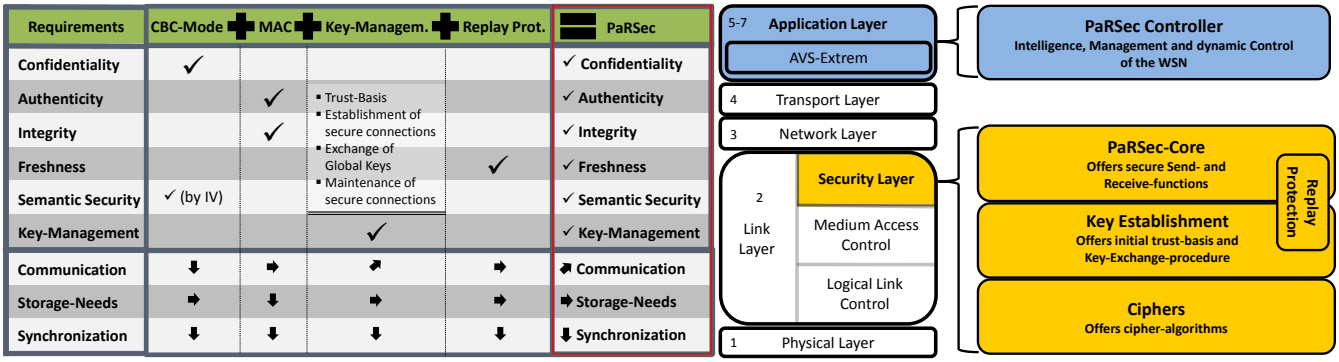

Figure 3: Composition of PaRSec (left), Integrating PaRSec into the ISO/OSI Network-Layer model (right)

propagated to the clients. In wired networks this part is usually covered by Public-Key or Diffie-Hellmann key-exchange protocols. Due to the known high resource constraints of WSNs these protocols are unfortunately too expensive. Thus, a protocol solely based on symmetric cipher-algorithms must be used. Although this problem has not been solved in a satisfying way yet [8], we have tailored an effective and innovative key-exchange protocol for the PaRSec system. Every sensor node starts out with an initial master key which is the same for all hosts. Since every sensor node needs to be flashed before deployment, this does not cause any additional effort. Based on this master key every node starts to establish pairwise keys with each of its 1-hop neighbors by broadcasting a HELLO-message containing an eight byte long nonce. Every node that receives this HELLO-message answers with an ACK-message containing its own nonce and a MAC calculated over both nonces to prove the correct reception of the nonce and the ownership of the correct master key. Once a node receives a nonce it will encrypt its own and the received nonce with the master key. The result of this encryption is used as the pairwise-key between the two nodes. Since only nodes in possession of the master key can do this and since all nodes (legally) in possession of the master key can be considered trustworthy, this new pairwise key can be considered secure. Once all nodes have established pairwise keys with all of their 1-hop neighbors, the base station will start to distribute a new global key by sending it to all of its 1-hop neighbors using the corresponding pairwise keys. All nodes receiving the new global key will distribute it in the same way, so that eventually the entire network will have the new global key. Since this new global key was generated by the secure base station and distributed over the secure pairwise keys, it can also be taken as secure. Since we now have a pairwise as well as a global secure connection there is no need for the initial master key anymore. Because of this very short time of usage of the initial master key, the chances of a successful attack on the key are negligibly small and therefore the entire protocol can be considered secure. To our knowledge, PaRSec is the first complete security system for WSNs, as it implements a Key Management in addition to all aforementioned security requirements.

5.3 Integrating PaRSec into AVS-EXTREM

The AVS-Extrem distributed event detection system is used to evaluate the applicability of our secure communication system in a deployment of a real-world fence monitoring system (see Figure 6). The whole event detection system consists of a system and an application layer as depicted in Figure 2. The system layer contains the AVS-Extrem-Board which is a 32-Bit ARM7 TDMI-S core based sensor node with 96 KB RAM and 512 KB Flash. The sensor node communicates at 868 MHz with the TI CC1101 transceiver. Sensors like the Bosch SMB380 acceleration sensor and the Sensirion SHT11 are integrated in the sensor board as well as a Molex SD-Card, a USB-connector, a JTAG connector and an external 12-pin connector for further modules. We use the FireKernel real-time OS [12] which features threading and priority based preemptive multitasking. FireKernel implements a reactive and dynamic routing protocol called Micro-Mesh-Routing that selects a robust path to the sink. The energy management fully implements Wake-On-Radio (WOR) and two kinds of energy saving modes of the ARM7 MCU (IDLE- and Power-down-Mode) to ensure an extended lifetime. The application layer contains the before mentioned distributed event detection and a Data Quality Estimator. The latter assesses the quality of the measured raw data as a preprocessing component of the subsequently used pattern recognition based distributed event recognition [5].

PaRSec integrates in the AVS-Extrem architecture in two parts, the PaRSec-Layer and the PaRSec-Controller as depicted in Figure 2. The PaRSec-Layer interacts with the system layer providing secure communications. The PaRSec-Controller takes care of the management and monitoring of the PaRSec-Layer.

5.4 Integrating PaRSec into ISO/OSI

The following section describes the architecture of PaRSec by the means of the placement in the ISO/OSI network layer model as well as the composition of the single components and their hierarchy.

One very essential question during the design of a security system is where to establish the new security (sub-)layer. Basically there are two possibilities: be-

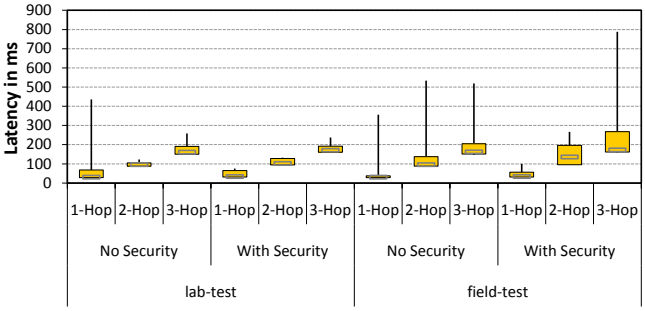

Figure 4: Results of the measured latency during the lab- and field-test

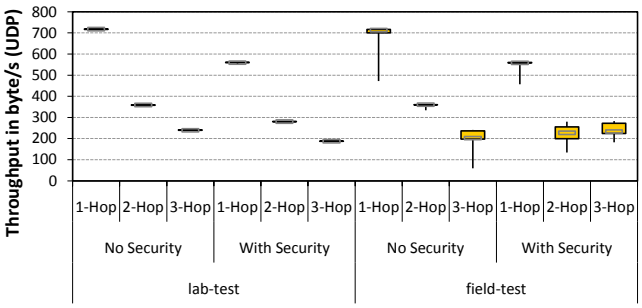

Figure 5: Results of the measured throughput during the lab- and field-test

Figure 6: Field-test with AVS-Extrem sensor nodes

low the network layer or at/above the network layer. Establishing it at or above the network layer has the downside of only having end-to-end connections available which leads to the problem that corrupted or manipulated packets can only be recognized as such at the destination node. The packet may thus travel over several hops and cause unnecessary traffic and energy-consumption before it is detected. In addition, the routing protocol which is typically settled at the network layer and for which numerous attacks exist can not be protected. This is because the packet-headers of underlying protocols and layers are added later on. The establishment at the link layer can prevent these problems. But there are some disadvantages as well. Since the link layer lies below the routing protocol, there is no routing protocol available for the security layer. Also, there is no reliable packet transportation protocol, e.g. TCP, available, so all these services need to be manually reproduced. As shown in Figure 3 (right), the decision fell in favor of the link layer. This is because attacks on the routing layer need to be prevented by any means while a missing routing protocol or reliable transportation can be overcome.

Since the PaRSec-Controller component does not access any functions of the system layer, it is settled at the application layer. The Cipher component of Figure 3 (right) contains the CBC-Mode and CBC-MAC implementation, as well as all available cipher-algorithms that can be used either individually or combined and are offered to the upper layers. Currently we support the cipher-algorithms: SkipJack, AES, RC5, TwoFish and 3DES where AES has outperformed the other algorithms in our experiments and was therefore selected to be the default one. Above the *Ciphers* component lies the *Key-Establishment* component which has already been described in section 5.2. The next higher component is the *PaRSec-Core* component which offers the secure send- and receive-functions needed by higher layers in order to use secure communications. As last of this part, the *Replay Protection* component spans over the *PaRSec-Core* as well as the *Key Establishment* component and takes care of the freshness requirement.

The second part of the software-architecture consists solely of the *PaRSec-Controller* component and is located at the application layer. This component manages the lower part of the architecture, i.e. it takes care of periodical global key updates, keep-alive messages, the own routing and also monitors the incoming packets for attacks.

This attack detection reports to the dynamic adjustment of the security-level manager which will decide how to react in the case of an attack or any suspicion of it. The PaRSec-Controller also offers different security presets, namely *No Security*, *Low Security*, *Optimal Security* and *Extreme Security* as well as *User Defined*. No Security disables the security layer while Low Security only ensures authentication. Optimal Security is the default preset and uses all security components and settings as recommended in this paper. Extreme Security combines all available cipher-algorithms for the en- and decryption of the packets. Our experiments always use Optimal Security.

6 EVALUATION

After having covered every security requirement, the distributed event detection applicability as an example for real-world applicability is left to be proven. The main metrics of the efficiency of the network are latency and throughput as well as energy efficiency. We evaluate the results of one lab- and one field-test in the following paragraphs.

6.1 Experiment-Setup

For the lab-test, we spread nine sensor nodes over a desk and set up several static routes over one to three hops. For the field-test we distributed 28 sensor nodes (see Figure 6) within our university building which in-

cludes several floors and two patios in a way that at least one 3-hop route was available. For this we applied the sensor nodes to doors, walls, and a dedicated construction site fence (7 nodes) to achieve a realistic scenario. For further details, see [11].

To evaluate the latency, the round-trip-time (RTT) of UDP-packets was measured over a time-span of 60 seconds per round. Each round was repeated 10 times for 1- to 3-hop routes. The average RTT divided by two is the final outcome in ms.

To measure the throughput, full packets were send to the destination which in turn acknowledged the reception and thereby started the next full packet. This was done over a time-span of 60 seconds per round. Each round was repeated 10 times for 1- to 3-hop routes. The accumulated number of payload *byte* divided by the time-span in seconds equals the throughput in $byte/s$.

6.2 Results

The results of these tests are shown in Figure 4 and 5. The results of the measurements using the security layer are compared to the results of the same test but without the security layer.

Lab-Test: For the latency we measured an average value of 34, 102 and 171ms for a 1- to 3-hops route respectively. The corresponding values without the security layer are 31, 90 and 101ms respectively. For the throughput we analogously measured 559, 281 and 187$byte/s$ with the security layer and 718, 359 and 239$byte/s$ without it. Overall this is a decrease of throughput of around 22% and an increase in latency of around 10% for the 1-hop case and around 6% for the 2- and 3-hop case.

Field-Test: Analogous to the syntax of the lab-test results the measured average latency during the field-test with the security-layer was 34, 117 and 173ms respectively. Without the security-layer the measured average latency was 31, 96 and 163ms respectively. As for the throughput the values were 536, 228 and 234$byte/s$ with security and 713, 360 and 202$byte/s$ without security. Noticeably the 3-hop value of the test without security is significantly lower than expected by inferring from the previous values. We investigated this matter and found out that the 3-hop route changed to a 2-hop route during the measurement. The cause of this can be found in the wireless medium and its changing interference characteristic due to moving people, weather conditions and other radio communication (external interference). In summary, this means a 10%, 22% and 6% increase of the average latency and a 25%, 37% and -15% decrease of the throughput in the 1-, 2- and 3-hop case respectively. During the field-test we also measured the time-span between triggering an event and the arrival of the corresponding alarm-message at the base station. The result was ten seconds with and without security which not only shows that the increase of latency really is negligibly small but that the entire distributed event-detection system still works very well with the security-layer enabled and thus the desired real-world applicability is achieved.

6.3 Energy Consideration

In addition to having the extra latency and lower throughput in realistic ranges, the (extra) energy consumption, also needs to be considered. There are basically two areas where additional energy consumption is caused by the security-layer: the en- and decryption of packets as well as the transmission and reception of longer or else a higher number of packets. To examine the impact of the security-layer on these areas, we measured the energy consumption that each cipher-algorithm needs to en- or decrypt 1 to 58$byte$ (maximum payload size of layer 2.5) and the energy needed to send a full packet with and without the security-layer (76 and 68$byte$ total respectively). Since the decryption executes the same operations as the encryption, the costs for decryption equal those of encryption. Also, in spite of our statement that the measurement starts with the encryption of one byte, actual values only start at 8 or even 16$byte$ onward. This is because the block cipher algorithms need a minimum number of bytes corresponding to the size of one block to execute a successful en- or decryption. As the 3DES algorithm consequently exhibited a ten- to a hundred-fold higher energy consumption than the other algorithms, we decided to not show it here. If we would have shown it, the scale of the y-axis would have been inappropriate for the remaining part of the plot.

Figure 7 shows the energy consumption of each cipher algorithm needed to encrypt from 1 to 58$byte$ of plaintext. As can be seen, the RC5 algorithm needed the fewest amount of energy ($0,004mJ$ for 8 *byte* up to $0,020mJ$ for 58$byte$) followed by AES, SkipJack and TwoFish. Again, the values of TwoFish nicely show the block oriented operation of the cipher-algorithms.

Figure 8 depicts the energy needed for the security-levels Low-Security (Auth_Only) using SkipJack and Optimal Security using the different algorithms each. The Low Security level only consists of a MAC-calculation while the Optimal Security level comprises the encryption of the payload as well as the MAC-calculation. Consequently, the Low Security level needs the fewest amount of energy. The order of the remaining results is analogous to those of Figure 7. Although in summary the RC5 algorithm seems to be the most efficient one, we disqualified it because it is patented and also known to have security-holes already. Since AES is the next most efficient algorithm (from $0,100mJ$ up to $0,206mJ$) and also a widely accepted standard, we chose it as the default algorithm for PaRSec. For comparison, we also added the amount of energy that the TI CC1101 transceiver of the AVS-Extrem Board requires to transmit one (full) packet. This puts a new perspective on the general rule of thumb for WSNs that the energy to transmit one packet outweighs the costs of every other operation by far [1]. Of course, this is also due to the relatively high-performance ARM7-MCU.

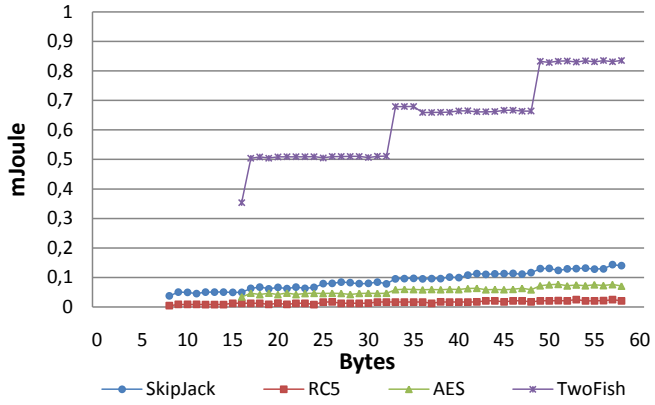

Figure 7: Energy needed by each algorithm for encryption of one to 58*byte* plain text

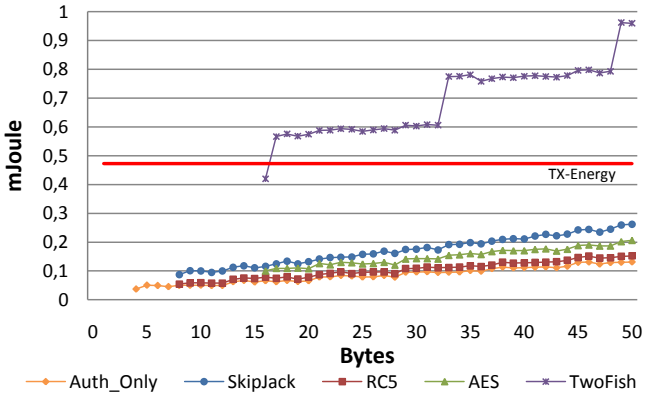

Figure 8: Energy needed for security levels with different cipher-algorithms (plain text encryption + MAC)

7 CONCLUSION

We have shown that the security requirements defined by the state of the art are either not completely covered by the currently available security systems for WSNs or the complete coverage causes prohibitively high additional expenses such that a realistic application of these systems is very unlikely. In the context of the distributed event-detection system project AVS-Extrem, a new layer 2.5 security system, PaRSec, was developed, integrated in the AVS-Extrem system and evaluated. For the first time, the complete coverage of all security requirements in a WSN security system is achieved, by utilizing efficient techniques that do not require any synchronization or state-keeping. The evaluation of PaRSec showed that there is a 10% increase of latency and a 25% decrease of throughput. It has also been shown that PaRSec did not have any significant negative impact on the functionality of the AVS-Extrem system and therefore PaRSec has proven to be real-world applicable. Since this has not been achieved by any security system before, PaRSec is a pioneering system and a great contribution to the scientific community.

Acknowledgements

This work was funded in part by the German Federal Ministry of Education and Research (BMBF, Project AVS-Extrem).

References

[1] Akyildiz, I., Vuran, M., Factors Influencing WSN Design. Ltd. 2010, [Online] John Wiley & Sons, Ltd., Available: http://dx.doi.org/10.1002/9780470515181.ch3

[2] Avancha, S., Undercoffer, J., Joshi, A., Pinkston, J., Secure sensor networks for perimeter protection. [Paper], Elsevier B.V., 2003.

[3] Carman, D., Kruus, P., Matt, B. 2000. Constraints and Approaches for Distributed Sensor Network Security. [Technical Report] Glenwood, MD, USA : NAI Labs, 2000.

[4] Çayırcı, E., Rong, C., Security in Wireless Ad Hoc and Sensor Networks. [Book] Chichester, United Kingdom : John Wiley & Sons, Ltd., 2009.

[5] Dziengel, N., Ziegert, M., Kasmi, Z., Hermans, F., Adler, S., Wittenburg, G., Schiller, J., A Platform for Distributed Event Detection in Wireless Sensor Networks. [CONET '10] Freie Universität Berlin, Germany April 2010.

[6] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K., System architecture directions for networked sensors. In Proc. of ACM ASPLOS IX, pages 93-104, November 2000.

[7] Karlof, C., Sastry, N., Wagner, D., TinySec: a link layer security architecture for wireless sensor networks. [SenSys '04] Baltimore, USA, 2004.

[8] Liu, D., Ning, P., Li, R., Establishing Pairwise Keys in Distributed Sensor Networks. [Paper] North Carolina State University, NC, USA, 2005.

[9] NIST, National Institute of Standards and Technology, DES Modes of operation. [FIPS 81] 1980.

[10] Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, J., SPINS: security protocols for sensor networks. [MobiCom '01] Rome, Italy, 2001.

[11] Schmittberger, N., Security in Event-Driven Wireless Sensor Networks. [Diploma-Thesis], Freie Universität Berlin, Germany, Feb. 2011.

[12] Will, H., Schleiser, K., Schiller, J., A real-time kernel for wireless sensor networks employed in rescue scenarios. [LCN '09], pages 834-841, New York, USA, October 2009.

[13] Wittenburg, G., Dziengel, N., Wartenburger, C., Schiller, J., A System for Distributed Event Detection in Wireless Sensor Networks. [IPSN '10], pages 94-104, Stockholm, Sweden, April 2010.

[14] Wittenburg, G., Dziengel, N., Wartenburger, C., Schiller, J., Verfahren und Sensornetz zur Merkmalsauswahl für eine Ereigniserkennung. German patent DE 10 2009 006 560 B4 (WO 2010/086325 A1), Jun. 2011.