Experimental Performance Evaluation of Cryptographic Algorithms on Sensor Nodes

Martin Passing and Falko Dressler
Autonomic Networking, Dept. of Computer Science 7, University of Erlangen-Nuremberg, Germany
martin@passing.de / dressler@informatik.uni-erlangen.de

Abstract— **Many security solutions have been proposed in the domain of wireless sensor networks (WSN). Usually, all these approaches are based on well-known cryptographic algorithms. The efficiency of WSN not only depends on effective routing decisions or energy aware wireless communication. In contrast, the efficiency strongly depends on the application scenario that includes appropriate security mechanisms. Previously, only few approaches were analyzed in experimental setups. Therefore, only theoretical measures were applied to demonstrate and discuss the behavior of security solutions. In this paper, we used an experimental setup for the verification of runtime behavior of several cryptographic algorithms including MD5, SHA-1, and AES. We used typical sensor hardware to get reasonable results. Based on our experiments, we provide some analysis and considerations on practical feasibility of such cryptographic algorithms in sensor networks.**

I. Introduction

In this paper, we study the behavior of wireless sensor nodes performing cryptographic operations. Instead of estimating the overhead introduced by such operations, we used a testbed to analyze the possible performance of typical cryptographic hash functions as well as of computational intensive encryptions. We figured out that most operations cannot be efficiently used on typical sensor nodes due to operating times of up to several seconds for a single operation. Therefore, the use of security functions in WSN must be carefully designed and evaluated even for simple setups.

Wireless sensor networks (WSN) have become a major research domain in the communications community [1]. Besides other issues that have been studied so far [2], energy consumption and security were identified to be the most challenging problem spaces. These properties are influenced by the massively distributed operating principle based on self-organization mechanisms [3]. Similarly, the lifetime of sensor networks [4] depends strongly on the operation mode, i.e. the used routing algorithms, the application behavior, and, finally, the employed security methods.

A survey of security issues in ad hoc and sensor networks can be found in [5]. Related work in the security area, focused on WSN, is summarized in [6]. We discuss additional work in section II. Obviously, there are numerous proposals for adding security features to WSN. Most of them concentrate on copying and adapting well-established approaches from Internet-based technology. Others propose new ideas but most of them have not yet outlined the feasibility of their proposals to work on real sensor nodes.

All approaches for enabling security in WSN are very scenario dependent. There are different requirements, for example, in an agriculture scenario [7] than in a habitat monitoring scenario [8]. Other requirements appear in the operation and control domain. Sensor nodes must be reconfigured, calibrated, and reprogrammed [9]. Such operations are very sensible for possible attacks. Finally, it must be mentioned that we ignore the problem of key management. Several solutions have been proposed that address this issue, e.g. [10].

In this paper, we present the results of an experimental evaluation of cryptographic operations using real sensor hardware. This study was inspired by previous work on security mechanisms for mobile systems such as mobile phones [11]. We implemented several cryptographic hash functions as well as an encryption operation for the BTnode sensors using available open source libraries. In several experiments, we evaluated the performance overhead due to such cryptographic functions.

The rest of the paper is organized as follows. In section II, we outline typical security architectures and the used cryptographic algorithms. Then, our implementation is briefly described in section III. A discussion of the measurement results follows in section IV. Finally, section V concludes the paper.

II. Security Solutions and Architectures

The purpose of this section is to summarize necessary information in the security domain that relates to this paper. We keep this section short because there are already good surveys available such as [5], [6]. The primary requirements on a successful security architecture are availability, authentication, data confidentiality, integrity, and non-repudiation. Most of these objectives can be addressed using cryptographic hash functions and appropriate encryption schemes. In ad hoc and sensor networks, many proposals were published concerning the use of security measures for particular applications [5]. Security protocols such as [6] define complex architectures to be used in a sensor network environment.

Most of such proposals defer the problem of key management - one of the most sophisticated problems - to be solved elsewhere. Fortunately, several approaches seem to be adequate in this domain. One example is the efficient public-key encryption in sensor networks [12]. A survey on key management solutions can be found in [13].

1-4244-0507-6/06/$20.00 ©2006 IEEE

Besides security architectures and special solutions for routing or key management, the aggregation of encrypted data in WSN was discussed [14] as well as the integration of particular security layers for reliable and secured communication [15]. Finally, secure overlays were proposed to address the security concerns in WSN [16].

In summary, it can be said that many promising proposals can be found in the literature that address the security objectives in sensor networks. Nevertheless, most of these papers only outline the principles or use simulation environments for verification. We tried to verify the applicability of such solutions on real sensor node hardware by analyzing the performance of several cryptographic algorithms.

In particular, we selected the following three cryptographic algorithms:

- MD5 (Message Digest) [17]
- SHA-1 (Secure Hash Algorithm) [18]
- AES (Advanced Encryption Standard) [19]

The first two algorithms, MD5 and SHA-1, represent cryptographic hash functions that are heavily used for typical message integrity checks and authentication. AES is a symmetric encryption algorithm that promises fast operation compared to asymmetric solutions. We selected these algorithms as candidates due to two reasons. First, these solutions are de facto standards in current security architectures and, secondly, open source implementations are available that can be used for our tests.

III. Experimental Implementation and Setup

A. Hardware and Software

As hardware platform for the tests, we used the BTnode sensor nodes developed at the ETH Zurich[1]. The nodes were used with the NutOS operating system and the BTnut system software version 1.6. In particular, we used the following libraries from the BTnut system software: thread, timer, event, terminal, ccc/bmac, Bluetooth, and eeprom. The BTnode architecture consists of an Atmel ATmega 128L micro controller operating at 8MHz with 4kByte flash and 244kByte RAM. For communication, a Zeevo ZV4002 Bluetooth system and a Chipcon CC1000 low power radio are available. An ISP and an UART interface are provided for programming and easy debugging, e.g. using a terminal program.

In addition to the standard system software, we used the following libraries to provide support for cryptographic algorithms:

- MD5/SHA-1: md5deep, version 1.12[2]
- AES: reference code version 2.2 by Rijndael[3]

We needed to customize these libraries for use in the node environment. Basically, only minor modifications were necessary such as providing appropriate integer data types (sometimes, 32 bit integer were assumed by the libraries).

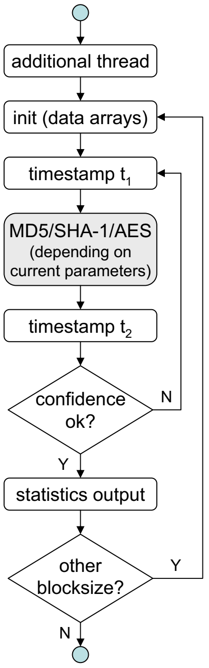

Fig. 1. UML diagram showing the main functions of the test setup

B. Implementation

In order to test the runtime behavior of the described cryptographic algorithms, we implemented a test environment as shown in figure 1.

First, a data-array is allocated with appropriate size, and filled with data (all bits zero/one, alternating values, etc.) to test the hash functions. The algorithm is given a pointer to this array and another to a data structure where it stores the resulting hash. For an encryption-algorithm a second data-array of same size is required to store the resulting en/decrypted data. After the execution the arrays are de-allocated.

For time measurement a 16-bit timer is used, which is being increased by the main system clock divided through 8 by a prescaler, resulting in a 1-μs-resolution. Furthermore, an IRQ-Handler is set which increases a global integer variable whenever the timer overflows. Through this the time since the activation of the timer can be measured exactly. To determine the time, which a hash or encryption algorithm needs, the system time before the computation is simply subtracted from the time thereafter.

To create arbitrary series of measured data, an outer loop was created that allows generating data-arrays with increasing size to be hashed or encrypted. For each step, either a single measure can be taken, or multiple measures for which statistical information such as the minimum, maximum, average, and median, are displayed, providing simplified further processing.

The interaction with the BTnodes is controlled by the terminal included in the 'BTnut System Software'. It provides the ability to execute any function and so start measurements or concurrent actions, and to extract the results immediately

[1] http://btnode.ethz.ch/

[2] http://md5deep.sf.net/

[3] http://csrc.nist.gov/encryption/aes/rijndael/

after the experiment using the text output.

For tests of effects due to concurrently running operations on the node, several additional tasks were implemented. First, another application was created, which can flood any node with data packets being sent through the Chipcon low-power radio. It can be run on a second node to verify the performance of the node under test while receiving data.

Currently, a standard BTnode has no sensors attached by default. Nevertheless, it is possible to measure the battery voltage which is similar to measuring sensor data. The 'BTnut System Software' provides a function to measure the voltage multiple times and return the average of it. This function can be run in another thread and periodically triggered through a timer to test the performance of the node when measuring and hashing or encrypting concurrently.

Another action that can be run in another thread is to send data packets of different size through the Bluetooth subsystem. This can also be triggered through a timer.

C. Experimental Setup

The experimental setup consisted of a PC running Linux (using a kernel providing USB-to-serial-support) and two BTnodes with attached connector boards to use USB connections to the PC. Using a terminal program, measurements and concurrent computations or data transmissions can be launched. The measurement results are then displayed in the terminal and can be copied to a file to be analyzed later.

IV. Measurements and Discussion

A. Executed measurements

In order to evaluate the performance of the cryptographic algorithms on the described sensor nodes, we executed the following measurements:

1) Hashing/encrypting data arrays of different size up to 1024 byte with MD5/SHA-1/AES. For each size, the test was executed multiple times and the median was calculated.
2) Hashing/encrypting a single array of 1024 byte with MD5/SHA-1/AES while using the Chipcon low-power radio:
   - ccc idle – with Chipcon enabled, idle
   - ccc receiving – with Chipcon enabled, receiving messages
   - ccc disabled – with Chipcon disabled
3) Hashing arrays of different size with MD5 while Chipcon low-power radio is enabled (idle)
4) Hashing 1024 byte with MD5 while periodically measuring the battery voltage
5) Hashing 1024 byte with MD5 while periodically sending packets through the Bluetooth subsystem

B. Results and Discussion

The plots depicted in figures 2 and 3 show the time needed to hash or encrypt data arrays of different size. For 1 kByte of data, hashing takes about 43 ms with MD5 and 129 ms with SHA-1. Encrypting with AES takes 1.67 s. The computation times are linearly dependent to the amount of data being processed. The MD5 algorithm needs the same time for data blocks of $n*64$ to $(n+1)*64-1$ byte. SHA-1 behaves similar to that, but the computation times even decrease a little within any of these intervals.

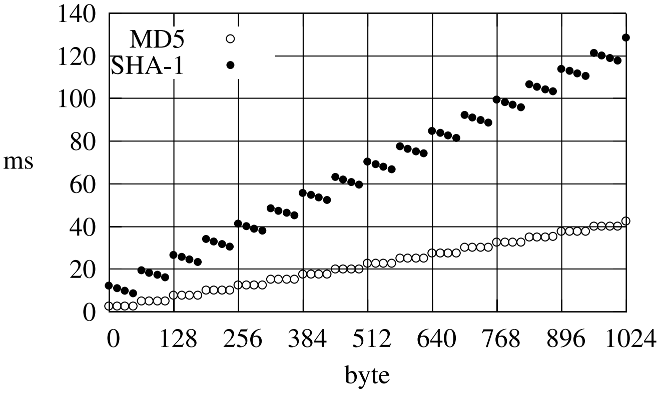

Fig. 2. Hashing arrays of different size with MD5 and SHA-1

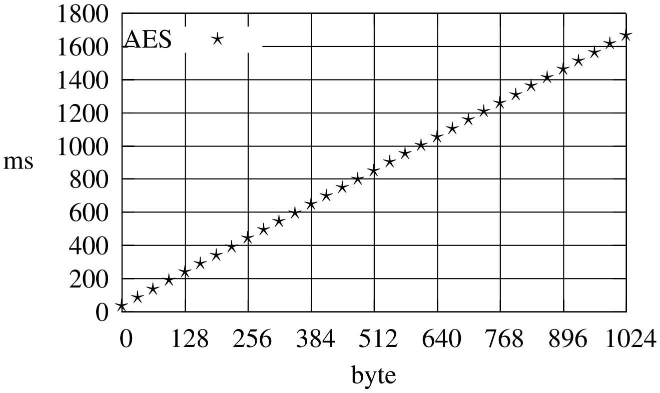

Fig. 3. Encrypting arrays of different size with AES

For the plots shown in 4, 5, and 6, measures of hashing or encrypting operations on a 1024 byte array were made. For each algorithm three series of single values were recorded: one with the Chipcon low-power radio disabled, one with the radio in idle mode, and one while concurrently receiving packets.

These plots show, that the computation times do not drift much when the radio is deactivated. When it is in idle mode, the times increase by a constant value and some additional peaks appear. These peaks seem to occur when the radio thread that periodically checks for incoming messages is triggered one more time during a single cryptographic computation. Then, the measured values are about 6200 μs higher, which seems to be the time needed to poll the radio.

When the node is flooded with data packets by another node using the Chipcon radio, the computing times alternate irregularly, sometimes being even smaller than before. This

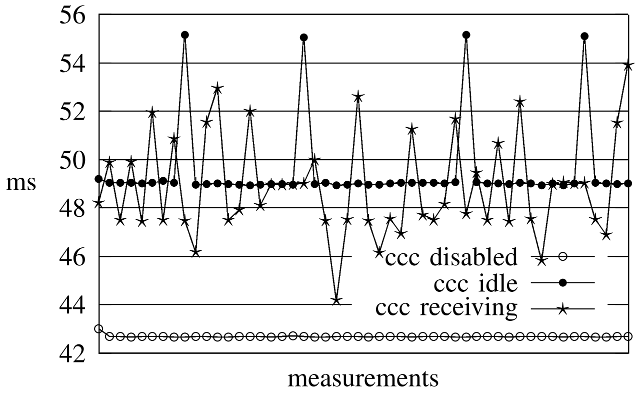

Fig. 4. Hashing 1024 byte of data with MD5 while using Chipcon radio

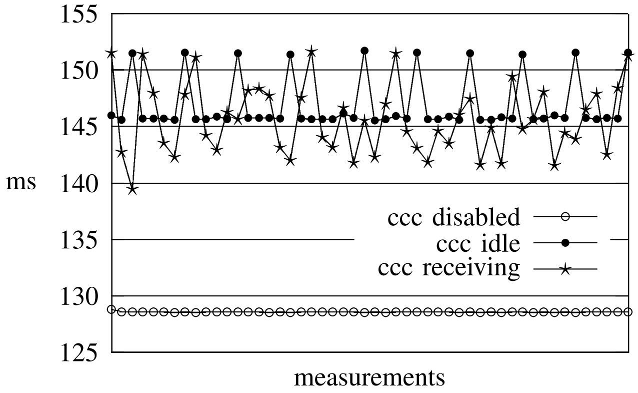

Fig. 5. Hashing 1024 byte of data with SHA-1 while using Chipcon radio

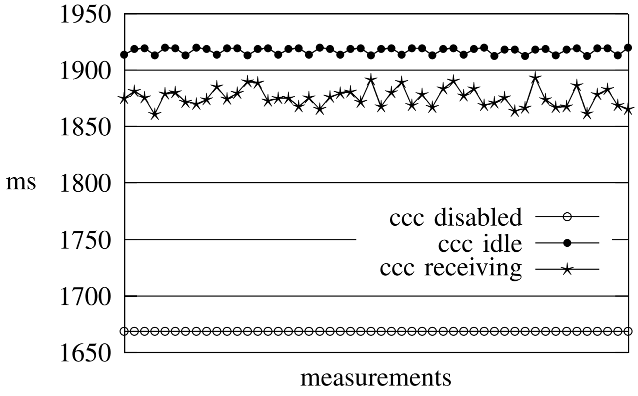

Fig. 6. Encrypting 1024 byte of data with AES while using Chipcon radio

seems to be due to the receiving thread returning faster when a packet arrives.

Tables I, II, and III depict also the minima, maxima, average, and median for all these measurements.

μs	disabled	idle	receiving
MIN	42673	48921	44188
AVG	42697.52	49499.4	48875.9
MED	42699	49014	48202
MAX	42995	55152	53916

TABLE I

HASHING 1024 BYTE OF DATA WITH MD5 WHILE USING CHIPCON RADIO

μs	disabled	idle	receiving
MIN	128496	145531	139450
AVG	128546.18	146876.48	145631.94
MED	128548	145723	145224
MAX	128817	151730	151635

TABLE II

HASHING 1024 BYTE OF DATA WITH SHA-1 WHILE USING CHIPCON RADIO

μs	disabled	idle	receiving
MIN	1668525	1668525	1912343
AVG	1668698.96	1668698.96	1917133.38
MED	1668696	1668696	1918802.5
MAX	1668722	1668722	1920002

TABLE III

ENCRYPTING 1024 BYTE OF DATA WITH AES WHILE USING CHIPCON RADIO

The measures shown in figure 7 were made on a BTnode with Chipcon radio activated and idling. They show the computing times of the MD5 algorithm being used multiple times for a data block of 64, 128, or 256 byte. Again, continuous peaks of 6200 μs appear, indicating the time consumed by the idling Chipcon thread waiting for messages, which is independent from the amount of data being hashed.

The plot in figure 8 shows the time needed to compute the MD5 hash of a 1024 byte array while the BTnode measures its battery voltage in another thread. If this happens 10 times a second, about half of the measurements take about 5500 μs longer. This is plausible as one computation takes 42700 μs on average, which is a little less than 1/20th of a second. If the battery measurement is triggered 50 times a second, this occurs two or three times during one computation. This behavior can be recognized easily in the plot.

With the Bluetooth subsystem activated, some measurements where made hashing 1024 byte with MD5 while another thread was sending packets. The results as depicted in figure 9 show that this did not influence the computations. This happened because the thread sending packets waited while the hashing algorithm was active, which could also be seen on the terminal.

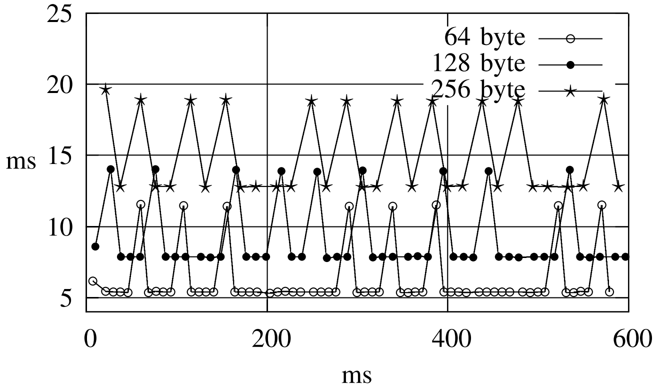

Fig. 7. Hashing arrays of different size with MD5 while Chipcon radio is idle

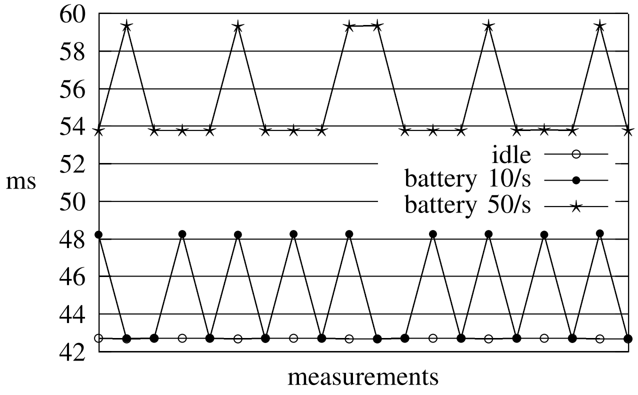

Fig. 8. Hashing 1024 byte of data with MD5 while measuring the battery voltage

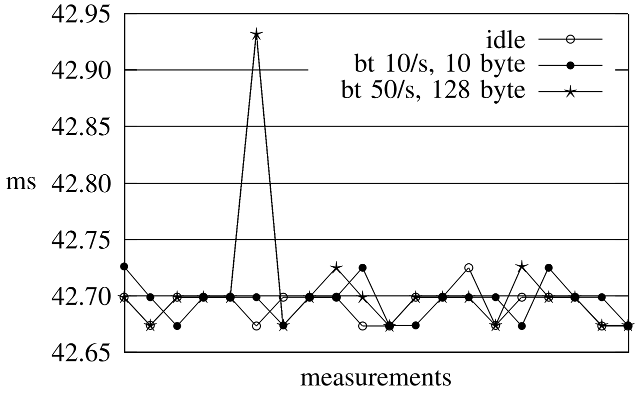

Fig. 9. Hashing 1024 byte of data with MD5 while sending data via the Bluetooth subsystem (bt)

V. Conclusion

We studied and analyzed several cryptographic algorithms using measurements on real sensor hardware. Such algorithms build the basis for all security solutions in communication networks including ad hoc and sensor networks. Focusing on the experimentation, produced statistically significant results by performing all measurements several times and analyzing singular effects as well as mean values (or more precise median values).

In conclusion, it can be said that the analyzed cryptographic algorithms need remarkably high execution times. For example, the execution of a single AES operation on a 1 kByte array lasted 1.67 s. Therefore, sensor nodes will need long times to perform adequate security operations for transporting several sensor measures. In scenarios as described in [14], multiple encryption/decryption operations are needed for data aggregation or in-network operation. We propose to use our measurement results as a basis for validating security scenarios for wireless sensor networks. Real measurements must build the basis for analyzing proposed security protocols in order to estimate their behavior. Additionally, our measurement results can be used to calibrate simulation setups in order to find out boundaries for real-time operation and communication.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–116, August 2002.

[2] C.-Y. Chong and S. P. Kumar, "Sensor Networks: Evolution, Opportunities, and Challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, August 2003.

[3] F. Dressler, "Self-Organization in Ad Hoc Networks: Overview and Classification," University of Erlangen, Dept. of Computer Science 7, Technical Report 02/06, March 2006.

[4] F. Dressler and I. Dietrich, "Lifetime Analysis in Heterogeneous Sensor Networks," in *9th Euromicro Conference on Digital System Design - Architectures, Methods and Tools (DSD 2006)*, Cavtat, Croatia, August/September 2006.

[5] D. Djenouri and L. Khelladi, "A Survey of Security Issues in Mobile Ad Hoc and Sensor Networks," *IEEE Communication Surveys and Tutorials*, vol. 7, no. 4, pp. 2–28, December 2005.

[6] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security Protocols for Sensor Networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, September 2002.

[7] A. Baggio, "Wireless sensor networks in precision agriculture," in *ACM Workshop on Real-World Wireless Sensor Networks (REALWSN 2005)*, Stockholm, Sweden, June 2005.

[8] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *First ACM Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, USA, September 2002.

[9] G. Fuchs, S. Truchat, and F. Dressler, "Distributed Software Management in Sensor Networks using Profiling Techniques," in *1st IEEE/ACM International Conference on Communication System Software and Middleware (IEEE COMSWARE 2006): 1st International Workshop on Software for Sensor Networks (SensorWare 2006)*, New Dehli, India, January 2006.

[10] W. Zhang and G. Cao, "Group Rekeying for Filtering False Data in Sensor Networks: A Predistribution and Local Collaboration-Based Approach," in *24th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2005)*, March 2005, pp. 503–514.

[11] T. Limmer, R. Gonzalez, and F. Dressler, "Real-time aspects on multiparty security on low-power mobile devices," in *Echtzeit im Alltag: Fachtagung der GI-Fachgruppe Echtzeitsysteme und PEARL (PEARL 2006)*, Boppard, Germany, November 2006.

[12] E.-O. Bla and M. Zitterbart, “Towards Acceptable Public-Key Encryption in Sensor Networks,” in *The 2nd International Workshop on Ubiquitous Computing (ACM SIGMIS)*, May 2005.

[13] X. Chen and J. Drissi, “An Efficient Key Management Scheme in Hierarchical Sensor Networks,” in *2nd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (IEEE MASS 2005): International Workshop on Wireless and Sensor Networks Security (WSNS’05)*, Washington, DC, USA, November 2005.

[14] C. Castelluccia, E. Mykletun, and G. Tsudik, “Efficient Aggregation of Encrypted Data in Wireless Sensor Networks,” in *Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2005)*, July 2005, pp. 109–117.

[15] F. Dressler, “Reliable and Semi-reliable Communication with Authentication in Mobile Ad Hoc Networks,” in *2nd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (IEEE MASS 2005): International Workshop on Wireless and Sensor Networks Security (WSNS’05)*, Washington, DC, USA, November 2005, pp. 781–786.

[16] H.-J. Hof, E.-O. Bla, and M. Zitterbart, “Secure Overlay for Service Centric Wireless Sensor Networks,” in *First European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004)*, August 2004.

[17] R. Rivest, “The MD5 Message-Digest Algorithm,” RFC 1321, April 1992.

[18] NIST, “Secure Hash Standard,” National Institute of Standards and Technology (NIST),” Federal Information Processing Standards Publication 180-1, April 1995.

[19] ——, “Specification for the Advanced Encryption Standard (AES),” National Institute of Standards and Technology (NIST),” Federal Information Processing Standards Publication 197, November 2001.