

# Realizing Robust User Authentication in Sensor Networks

Zinaida Benenson\*  
Chair of Computer Science 4  
RWTH Aachen University  
zina@i4.informatik.rwth-  
aachen.de

Nils Gedicke\*  
Chair of Computer Science 4  
RWTH Aachen University  
nils.gedicke@post.rwth-  
aachen.de

Ossi Raivio  
Chair of Wireless Networks  
RWTH Aachen University  
ora@mobnets.rwth-  
aachen.de

## ABSTRACT

We investigate how to organize access control to the WSN data in such a way that an unauthorized entity (*the adversary*) cannot make arbitrary queries to the WSN. We call this problem *authenticated querying*. Roughly, this means that whenever the sensor nodes process a query, they should be able to verify that the query comes from a legitimate user. Authenticated querying is especially challenging if the adversary can gain full control over some sensor nodes through physical access (*node capture* attack). We propose first solution to this problem and present our experiments with an implementation of the first step of this solution.

## 1. INTRODUCTION

Wireless sensor networks (WSN) gather environmental data, process and store them, and finally give these data to the user, either on demand or upon event detection. This means that sensor networks offer services to the users.

Consider a sensor network spread over a large geographic area. The maintainer of the sensor network offers services to a large number of mobile users. For example, sensor readings could be used for precision agriculture. Farmers can subscribe to the services and remotely query sensors on their fields using some mobile device like a PDA. In this case, only the queries of a legitimate user should be answered by the network. However, existing query processing systems (for an overview, see e.g. [18]) are not concerned with this issue. Meanwhile, this problem becomes especially difficult in presence of node capture attacks.

*Node capture* means gaining full control over a sensor node by a physical attack, e.g., opening the sensor's cover and appending some wires somewhere. This type of attack is fundamentally different from gaining control over a sensor remotely through some software bug, e.g., a buffer overflow. As all sensors are usually assumed to run the same software, in particular, the same operating system, finding an appropriate bug would allow the adversary to control the whole sensor network. In contrast, a node capture attack can only be mounted on a small portion of a sufficiently large network.

In presence of node captures, critical tasks such as posting

\*Zinaida Benenson and Nils Gedicke were supported by German Research Foundation (DFG) as part of the Graduate School "Software for Mobile Communication Systems" at RWTH Aachen University.

queries to the WSN, cannot rely on a single sensor, as otherwise a captured sensor could make arbitrary queries on behalf of the adversary, or give fake data to a legitimate user. In previous work [2] we identified and formalized data access issues in this adversary model.

## Contributions

We identify a new security issue for WSNs — the *authenticated querying* problem. We propose first, to our knowledge, solution to this problem in presence of node capture attacks.

We report our experience with an implementation of the first step of this solution, a robust user authentication mechanism. The user needs to communicate with several sensors concurrently before he receives access to the WSN data. We investigated the feasibility of this approach for WSNs consisting of Telos Revision B nodes [11] which run TinyOS [16].

Our user authentication protocol is based on public key cryptography which is said to be feasible for WSNs even without special hardware support [17, 9]. To verify these claims, we implemented digital signatures with the EccM library for elliptic curve cryptography by D. Malan [9]. From the logical point of view, public key crypto is more appropriate for our task than symmetric key cryptography, see Section 2.3 for the corresponding reasoning. To our knowledge, this is a first attempt to implement an advanced authentication protocol using the EccM library.

## Roadmap

In Section 2, we present the authenticated querying problem. After outlining our system model (Section 2.1), we define authenticated querying (Section 2.2) and outline a possible solution to this problem (Section 2.3). In Section 3 we present our initial implementation of this solution. We first give the protocol specification (Section 3.1), then the experimental results (Section 3.2). Analysis of the protocol is presented in Section 4. We refer to related work in Section 5 and conclude in Section 6.

## 2. AUTHENTICATED QUERYING

### 2.1 System and Adversary Model

We assume a large static sensor network. The users can access the network using some mobile device. The users are mobile, but during a particular querying process they have to remain in place. On average, there are  $n$  sensors

in the communication range of the user. Of these,  $t$  sensors are allowed to fail or to be malicious, meaning that they are captured and run programs which are different from the expected ones. For example, they can falsely authenticate illegitimate users, deny access to legitimate ones, or alter data which they are supposed to send to the user. Therefore, the user can rely for communication on at most  $n-t$  sensors in his communication range.

## 2.2 Problem Statement

In general, whenever sensor nodes process a query, they should be able to verify that the query comes from a legitimate user. We call this problem *authenticated querying*. More formally, a WSN enables authenticated querying if it satisfies the following properties (perhaps, with some large probability):

- *Safety*: If a sensor  $s$  processes the query  $q$ , then  $q$  was posted by a legitimate user  $U$ .
- *Liveness*: Any query  $q$  posted by a legitimate user  $U$  is processed by at least all sensors  $s \in S_q$ , where  $S_q$  is the set of sensors which must process the query in order to give the required answer to the user.

Next section outlines one of possible solutions to authenticated querying in presence node capture attacks.

## 2.3 An Idea for Authenticated Querying

If the number of users is large, the natural method to use for authentication is public key cryptography because of its scalability. On the other hand, public key cryptography is power-hungry, so the sensors should communicate with each other using symmetric cryptography. Our concept is to let the sensors in the communication range of the user serve as interpreters (or a gateway) between the “public key crypto world” of the user and the “symmetric crypto world” of the WSN. The user talks to sensors in his communication range using public key cryptography, and these sensors then talk to the remainder of the sensor network on behalf of the user using symmetric cryptography. This happens in authenticated fashion:

1. *Robust secure channel setup between the user and the WSN*: The user executes a mutually authenticated key establishment protocol [10] using public key cryptography with a specified number of sensors in his communication range. The protocol results in establishment of symmetric session keys between the user and each correct sensor which participated in a protocol run.
2. *Authenticated query forwarding*: After the successful secure channel setup, the sensors in user’s proximity forward user’s query into the sensor network and append to it some additional information which enables the other sensors to verify the legitimacy of the query.

We briefly outline a possible solution to authenticated querying assuming that the query is addressed to a single sensor  $s$ . The user first sends his query to the surrounding sensors

using secure channels. Then the sensors in user’s communication range compute symmetric keys which they share with the sensor  $s$  [8]. Each sensor computes message authentication code (MAC) on the query. These MACs are sent back to the user who appends them to the query. The sensor  $s$  answers the query only if enough MACs are appended.

Note that in this solution, no coordination between the sensors in user’s proximity is required. They do not need to know  $n$  or  $m$ . In case an entity is unable to authenticate to  $m$  sensors, not enough MACs are appended to the query, and therefore, the query is rejected by sensor  $s$ . Solutions where no coordination between the sensors is needed should generally be preferred. Otherwise, coordination requires additional messages, and therefore, additional resource consumption.

The above solution explains the idea and shows the feasibility of our approach. However, it has many drawbacks. For example, the legitimacy of the query cannot be verified by intermediate sensor nodes which route the query to the sensor  $s$ . This feature would enable early rejection of illegitimate queries. We consider more practical and sophisticated solutions in a concurrent work [1].

## 3. A PROTOCOL FOR ROBUST USER AUTHENTICATION

To verify the feasibility of the above idea for authenticated querying, we partially implemented the first stage of the solution. In our implementation, the user unilaterally authenticates to the sensors in his proximity using public key cryptography. We call this part of solution *robust user authentication*.

Mutual authentication and session key establishment are left to the future work.

### 3.1 Specification

#### 3.1.1 Choice of a Public Key Cryptosystem

RSA with small public exponent ([7, 17]) and Rabin public key cryptosystems ([5]) have fast algorithms for encryption and digital signature verification. However, decryption and signature generation are slow and resource-demanding. Therefore, these cryptosystems can be used in sensor networks only if the sensors are not required to decrypt or to sign messages.

In contrast, elliptic curve cryptosystems (ECC) [7, 9] require more overhead for encryption and signature verification than for decryption and signing. Nevertheless, with ECC, not only encryption and signature verification, but also decryption and signing are feasible for sensor nodes. As our full solution targets mutual authentication, we chose ECC for implementation of robust user authentication, even though the sensors in this case have to execute relatively expensive for ECC signature verification.

#### 3.1.2 System Setup and Assumptions

We consider a public key infrastructure (PKI) for elliptic curve cryptography. There is a certification authority  $CA$  in the system. This could be the base station. The  $CA$  has a private/public key pair ( $priv\_key_{CA}, pub\_key_{CA}$ ) and gives to

each legitimate user a certificate. The certificate is the user’s public key signed by the *CA*:  $cert_U = sign_{CA}(pub\_key_U)$ .

Each sensor has the public key of the *CA* preloaded. With this public key, each sensor can verify the certificate. We used the EccM library [9] to implement a digital signature method with message recovery called the Nyberg-Rueppel scheme [14]. “Message recovery” means that upon signature verification, the verifier also gets the signed message. Therefore, the certificate verification process also extracts the public key of the user from the certificate.

Standard certificates include, in addition to a public key signed by the certification authority, also at least user’s name and the expiration date [10]. However, we propose a different approach for sensor networks. As time synchronization is a difficult issue in WSNs, and Telos motes do not have a real-time clock, we propose a different certificate management strategy. The certificate consists only of a signed public key of the user. The public key of the certification authority is changed periodically and distributed to all sensors in the WSN. This can be done, e.g., daily or monthly, depending on the application. Any entity which possesses a public key signed with the current key of the certification authority, can access the network. The users could refresh their keys at the web site of the WSN maintainer.

### 3.1.3 Parameters

- $n$  sensors (on average) in the communication range of the user, denoted as  $s_1, \dots, s_n$
- $t < n$  sensors can be captured and run programs which are different from the correct, authorized one.
- $t < m \leq n - t$  is the number of sensors which must successfully authenticate the legitimate user, and deny access to illegitimate one. W.l.o.g. we denote these sensors as  $s_1, \dots, s_m$ . For example, if  $n = 10$  and  $t = 3$ , we could set  $m = 6$ .
- For the calculations concerning the elliptic curves we use the parameters hardcoded in the original EccM library by David Malan. With these parameters, it is possible to manage 163 bit keys which is a sufficiently large value for a secure authentication algorithm.

It is worth mentioning that 163 bits is the size of a private key. Public keys consist of two 163-bit numbers, and Nyberg-Rueppel digital signature on a 163-bit message also consists of two 163-bit numbers. Therefore, a certificate, which is basically a signed by a certification authority public key, consists of four 163-bit numbers.

In the following, we describe our protocol. It is based on a standard challenge-response protocol with digital signatures [10].

### 3.1.4 Protocol

1.  $U \Rightarrow WSN: (U, cert_U)$

The user stands somewhere in the sensor field and starts the protocol. He first broadcasts his identity and certificate to the sensors in his communication range.

2.  $\forall 1 \leq i \leq m: s_i \rightarrow U: (s_i, nonce_i)$

Upon receiving user’s first message, each sensor  $s_i$  saves it and sends his identity and a random challenge  $nonce_i$  to the user. “Nonce” means “number used only once” and is a standard term from the cryptography.

Because the first message is broadcast, all the sensors hear it practically simultaneously. If all the sensors reply immediately, this will result in collisions. Therefore, a random backoff is needed.

The medium access control (MAC) layer in Telos motes [12] is practically the same as B-MAC [13] which has been part of TinyOS already for some time. When there is data to be sent, medium is detected. If the medium is free, random backoff of  $(0, 320, \dots, 4800)$   $\mu s$  is used. After this period, the medium is detected again. If it is free, packet is sent. If the medium is busy, congestion backoff of  $(0, 320, \dots, 20160)$   $\mu s$  is used. There are at most eight retries.

In our tests the backoffs provided by the operating system were not enough and at worst all the packets were lost due to collision. Because of this, we use a random backoff of  $(0, 1, \dots, 255)$  ms before trying to send the response to the broadcast.

3.  $\forall 1 \leq i \leq m: U \rightarrow s_i: sign_{U_i}(h(U, s_i, nonce_i))$

The user answers the challenge to each sensor. He first constructs a hash  $h(U, s_i, nonce_i)$  using the hash function  $h$ . In our implementation, we used the SHA-1 hash function [15] which provides 160-bit hash values. Here, the identity of the user  $U$ , the identity of the sensor  $s_i$  and the nonce are concatenated before hashing.

4.  $s_i: verify(cert_U) := pub\_key_U$   
 $s_i: verify(sign_{U_i}(h(U, s_i, nonce_{s_i})))$

Each sensor  $s_i$  extracts user’s public key  $pub\_key_U$  from the certificate  $cert_U$ . It then uses the extracted public key to extract the content of the third protocol message which is supposed to be  $h(U, s_i, nonce_i)$ . Then the sensor independently computes the hash value and compares it to the extracted one. The user is successfully authenticated if and only if the values are equal.

## 3.2 Experimental Results

### 3.2.1 Implementation details

We implemented our protocol on Telos Rev. B motes [11] using the EccM library for elliptic curve arithmetic [9]. The algorithms to sign a message and to verify a signature were implemented by combining and modifying the functions of EccM. Nevertheless, some more functionality was needed like a modular multiplication function and the procedure to generate the SHA1 hash. The whole program uses 45.5 kB of ROM and 2.0 kB of RAM.

### 3.2.2 Protocol execution with one mote

The charge and time consumption for one mote is shown in Table 1. It is based on the measured execution time of the implementation and nominal figures for power consumption from [11]. Absolute majority of overall time and sensor’s energy are consumed by its calculations. The user’s energy

Step	No. of packets	Time [s]		Charge [mC]	
		$U$	$s$	$U$	$s$
1	9	0.011	0.011	0.24	0.24
2	1	0.001	0.001	0.03	0.03
3	3	65	0.004	117	3.6
4	1	0.001	375	20	675

**Table 1: Time and charge consumption in the case of a user ( $U$ ) and one sensor ( $s$ ). Number of packets contains both the received and sent data packets and acknowledgments.**

is mainly consumed in signing operation. Radio communication takes only a minute part of the whole.

One authentication takes approximately 440 s and consumes 140 mC on user side and 680 mC on sensor side. A good battery contains a charge of 1500 mAh. If not doing anything else, a sensor can run our authentication cycle around 8 000 times and user 40 000 times before exhausting their energy sources.

As our user also was a sensor, instead of being a more powerful device like a PDA or a mobile phone, execution time and energy consumption in step 3 (user signs a nonce) would be much lower in real life.

### 3.2.3 Protocol execution with several notes

We performed several test runs with several combinations of  $1 \leq n \leq 5$  and  $1 \leq m \leq 5$ , where  $n$  is the number of sensors in user's communication range, and  $m$  is the amount of sensors required to authenticate the user. Because our conditions were near-ideal the system behaved as it theoretically should have behaved: no packet losses or deadlocks were encountered, legitimate user was never denied access in the presence of enough sensors. Thus, the initial results seem promising. However, more exhaustive testing with e.g. radio interference, longer distances, absence of line of sight, and specifically tailored bogus messages are needed in order to be assured of the protocol's robustness.

## 4. PROTOCOL ANALYSIS

### 4.1 Impersonation Attacks

As in our implementation the sensor nodes do not authenticate to the user, a single captured sensor node could impersonate  $m - 1$  valid sensor nodes and authenticate the adversary. However, in a fully implemented solution to authenticated querying, sensor nodes also authenticate to the user, and append some information to the query, such that other sensors can verify the legitimacy of the query. Therefore, care should be taken that the proof of query legitimacy cannot be produced by less than  $m$  sensors (perhaps, with some large probability).

For example, in the solution outlined at the end of Section 2.3, enough MACs should be appended to the query. An adversary which captured less than  $m$  sensors, would not be able to produce  $m$  valid MACs needed for authentication information.

### 4.2 Denial-of-Service Attacks

Denial-of-service (DoS) attacks are most difficult to mitigate, as they target the inherent resource constraints in WSNs [19].

The easiest DoS attack on wireless communication is jamming. Besides, as in Steps 2 and 3 the user communicates with at least  $m$  sensors concurrently, DoS by deliberate collisions on MAC layer should be taken into account. Some defenses are presented in [19].

As sensors save user's certificate, there is a possibility for DoS attacks by broadcasting several bogus certificates in Step 1 of the protocol. The sensor then would have to save them (and the nonces in Step 2) and thus sensor's memory could be exhausted. Therefore, we do not allow a sensor to run authentication protocols with multiple users concurrently. This does not seem to be a severe restriction in our setting, as we do not expect several users to be in the same location anyway.

However, there is a very severe DoS possibility against which no strong defense is known. If an adversary sends a bogus certificate in Step 1 and then a bogus signature in Step 3, the only possibility to discover this attack for a sensor is to verify both the certificate and the signed challenge. As the signature verification takes up very much energy and time, the sensor nodes would run out of energy rather quickly.

To mitigate this attack, some redundancy could be added to user's certificate, such that the sensor could discover the attack immediately after certificate verification. However, this is a very weak defense: The adversary could eavesdrop on the first protocol step of a legitimate user and then replay the valid certificate to sensors in an arbitrary part of the WSN.

Such an attack would deny access to a legitimate user, but would not give the adversary unauthorized data access. Perhaps the most promising procedure for defense would be by combining technical and procedural means. The gain to the adversary in service denial to legitimate users should be reduced, e.g., by data replication techniques. On the other hand, measures for quick response should be developed, such as quick deployment of additional sensors in the affected area and notification of the WSN about the occurred attacks, such that new attacks could be quickly discovered by the forewarned sensors and reported to the base station.

## 5. RELATED WORK

In the last few years, quite a number of papers considered implementing efficient public key cryptography on small devices. Most relevant for the WSNs are [7, 9, 17]. Most related to our work is the EccM library [9] which we used in our implementation.

As for advanced cryptographic protocols, D. Malan et al. implemented Diffie-Hellman key exchange with elliptic curve cryptography on MICA2 motes [9]. R. Watro et al. [17] implemented an RSA-based authentication protocol on MICA2 motes where the user authenticates to each sensor node separately in order to access sensor's local readings. And finally, after our paper was submitted, a great breakthrough

was reported by Gupta et al. [6]. They implemented the SSL protocol on MICA2 motes using elliptic curve cryptography. This confirms choice of ECC for our implementation and the feasibility of mutually authenticated key establishment on sensor nodes, as their SSL handshake takes less than 4s.

However, neither of above papers considers building an access control mechanism which can withstand node capture. To our knowledge, we are the first to consider authenticated querying in WSNs in presence of node capture attacks.

## 6. CONCLUSIONS AND OUTLOOK

We implemented a user authentication protocol which is robust against node capture attacks. The protocol is based on elliptic curve cryptography and utilizes redundancy to withstand node capture. This protocol is our first step towards realizing authenticated querying in WSNs. This means that whenever the WSN processes a query, sensor nodes should be able to verify that the query comes from a legitimate user. As there are potentially many users in the WSN, we use public key cryptography for user authentication, as it scales much better than symmetric cryptography approaches. The basic idea is to use sensors in user's proximity as interpreter between the "public key crypto world" of the user and the "symmetric crypto world" of the WSN.

The protocol execution time of several minutes is disappointing, which is mainly due to long execution times of elliptic curve cryptography routines in EccM. EccM needs 34 sec for one point multiplication on MICA2 motes [4], which accounts for long execution time for signature generation and verification in our protocol. However, in [6] execution time of less than 4s for full SSL handshake is reported (in assembly language). Also in *nesC* more efficient execution times are now being implemented according to [9] and to personal communication with E.-O. Blaß (see also [3]). Therefore, our ideas are becoming applicable.

## Acknowledgments

We thank one of the anonymous reviewers for detailed analysis of our approach which greatly helped to improve the paper.

## 7. REFERENCES

- [1] Z. Benenson. Authenticated queries in sensor networks. (in submission).
- [2] Z. Benenson, F. C. Gärtner, and D. Kesdogan. An algorithmic framework for robust access control in wireless sensor networks. In *Second European Workshop on Wireless Sensor Networks (EWSN)*, January 2005.
- [3] E.-O. Blaß and M. Zitterbart. Towards acceptable public-key encryption in sensor networks. In *The 2nd International Workshop on Ubiquitous Computing*, May 2005. to appear.
- [4] Crossbow, Inc. MICA2 data sheet. Available at [http://www.xbow.com/Products/Product\\_pdf\\_files/Wireless\\_pdf/MICA2\\_Datasheet.pdf](http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf).
- [5] G. Gaubatz, J.-P. Kaps, and B. Sunar. Public key cryptography in sensor networks - revisited. In *ESAS*, pages 2–18, 2004.
- [6] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, and S. C. Shantz. Sizzle: A standards-based end-to-end security architecture for the embedded internet. In *Third IEEE International Conference on Pervasive Computing and Communication (PerCom 2005)*, Kauai, March 2005.
- [7] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In *CHES2004*, volume 3156 of *LNCS*, 2004.
- [8] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 52–61. ACM Press, 2003.
- [9] D. J. Malan, M. Welsh, and M. D. Smith. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In *First IEEE International Conference on Sensor and Ad Hoc Communications and Network*, Santa Clara, California, October 2004.
- [10] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, 1997.
- [11] Moteiv, Inc. Telos revision B datasheet. Available at <http://www.moteiv.com/products/docs/telos-revb-datasheet.pdf>.
- [12] J. Polastre and A. Broad. Module for Chipcon 2420 series radio in TinyOS. [Source code.] Available at <http://www.tinyos.net/tinyos-1.x/tos/lib/CC2420Radio/>.
- [13] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA, 2004. ACM Press.
- [14] M. Rosing. *Implementing elliptic curve cryptography*. Manning Publications Co., Greenwich, CT, USA, 1999.
- [15] Secure Hash Standard. FIPS PUB 180-1, April 1995. Available at <http://www.itl.nist.gov/fipspubs/fip180-1.htm>.
- [16] TinyOS web site, <http://www.tinyos.net/>.
- [17] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus. TinyPK: securing sensor networks with public key technology. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 59–64. ACM Press, 2004.
- [18] A. Woo, S. Madden, and R. Govindan. Networking support for query processing in sensor networks. *Commun. ACM*, 47(6):47–52, 2004.
- [19] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *Computer*, 35(10):54–62, 2002.