Authenticated Query Flooding in Sensor Networks

Zinaida Benenson[1], Felix C. Freiling[2], Ernest Hammerschmidt[1]
Stefan Lucks[2], and Lexi Pimenidis[1]

[1] Department of Computer Science, RWTH Aachen University, Germany
{zina,ernest,lexi}@i4.informatik.rwth-aachen.de
[2] Department of Computer Science, University of Mannheim, Germany
{freiling, lucks}@uni-mannheim.de

Abstract. We propose a novel mechanism for authentication of queries in a sensor network in case these queries are flooded. In our protocol, the base station appends an authenticator to every query, such that each sensor can verify with certain probability that the query is sent by the base station. Implicit cooperation between sensor nodes during the flooding process ensures that legitimate queries propagate quickly in the network, whereas the propagation of illegitimate queries is limited to only a small part of the network.

1 Introduction

Wireless sensor networks consist of a large amount of sensor nodes, which are small low-cost wireless computing devices equipped with different sensors. They measure and collect environmental data. Access to these data is organized via a special gateway, called base station, which sends queries into the sensor network and gives the required sensor data to the users.

According to some paradigms for organizing sensor networks, such as Directed Diffusion [11] and TinyDB [14], the base station floods the sensor network with the query, as identifiers and locations of sensor nodes which are able to answer a particular query are not known to the base station beforehand. An example of such a query is "which sensor nodes measure temperature above θ grad?".

As data gathered by the sensor network may be valuable or critical, it should be protected from the unauthorized access. In particular, only the base station should be allowed to send queries. In this work, we consider how the base station can authenticate its queries, such that only legitimate queries are answered by the sensor nodes.

Our idea is based on the fact that in general, before the query reaches the nodes which are able to answer it, it has to be flooded through some significant part of the network. Thus, if some mechanism allows the sensor nodes to verify the legitimacy of the query probabilistically, illegitimate queries will be dropped before they get deeply into the network and reach their target nodes.

We developed a probabilistic protocol which restricts propagation of fake queries to a logarithmic part of the network. Some sensor nodes in our protocol may fail to recognize a fake query. This does not matter much, however, as long as their number is "very small". We think, logarithmic part of the network qualifies for the term "very small", considering that sensor networks may consist of thousands and hundred thousands of nodes.

Please use the following format when citing this chapter:
Author(s) [insert Last name, First-name initial(s)], 2006, in IFIP International Federation for Information Processing, Volume 201, Security and Privacy in Dynamic Environments, eds. Fischer-Hubner, S., Rannenberg, K., Yngstrom, L., Lindskog, S., (Boston: Springer), pp. [insert page numbers].

Roadmap We review related work in Section 2. We define Authenticated Query Flooding (AQF) in Section 3, and present our basic protocol `AQF-pass`, accompanied by theoretical analysis and simulation results, in Section 4. In Section 5, we discuss possible improvements to the basic protocol, and overview our ongoing and future work.

2 Related Work

One possibility for an entity to authenticate its messages to multiple receivers is *authenticated multicast* (or *broadcast*). The receivers can verify the origin of the message using some attached authentication information, called *authenticator*. However, no receiver can generate the authenticator, and therefore, cannot impersonate the sender.

Some approaches to authenticated broadcast in sensor networks exist in the literature. In SPINS [18], authenticated streaming multicast μTESLA is realized using one-way hash chains, time synchronization, and symmetric keys shared by the base station with each sensor in the network. μTESLA is a very efficient protocol. Its security depends on the security of the underlying time synchronization mechanism. However, devising a protocol which globally synchronizes time in a large sensor network seems to be a difficult problem [7].

Relatively inexpensive digital signatures can also be used for authenticated flooding (see, e.g., [20]), assuming that each sensor node is preloaded with the public key of some certification authority. However, these signatures are still very expensive considering the limited resources of sensor nodes.

Our protocol uses only symmetric cryptography. The protocol is based on the ingenious protocol by Canetti et al. [3], but it has a much better performance, as our protocol relies on the implicit cooperation between the sensor nodes which occurs when the authenticated query is flooded into the network.

3 System Model

Sensor Network Architecture Sensor network is spread over a large geographic area and consists of homogeneous sensor nodes which are similar to Telos sensor nodes [16] in construction and performance. They may have 8 or 16 bit microcontroller, with the amount of RAM varying between 2 kB and 10 kB and flash memory ranging from 48 kB to 128 kB. The speed of radio communications is in the order of 100 kbps.

There is also a base station in the network which is a laptop class device. The base station is trusted by sensor nodes. It can flood queries into the sensor network. The queries sent by a base station are called *legitimate queries*.

Adversary Model The adversary is an illegitimate entity interested in the data produced by the sensor network. The goal of the adversary is to post *arbitrary* queries to the sensor network, just like the base station can do. The queries sent by the adversary are called *illegitimate* (or *fake*) *queries*.

The adversary can capture some sensor nodes. Capturing means gaining information from a sensor node through direct physical access. In this case, we assume that the adversary knows the cryptographic keys of the captured nodes. As discovered in [1],

node capturing requires non-negligible amount of time and resources. Therefore, we assume that the adversary can capture only a small amount of sensor nodes, in order of tens, but certainly not of hundreds.

Definition of Authenticated Query Flooding Let WSN be a sensor network. After receiving a query, each sensor node *decides* whether the query comes from the base station. If its decision is positive, we say that the sensor *accepts* the query. Consider an arbitrary query q. The WSN design satisfies *authenticated query flooding (AQF)* if it satisfies the following properties:

- (Safety) If a sensor s in WSN accepts the query q as a legitimate query, then q is a legitimate query, e.g., q was originated by the base station.
- (Liveness) Any legitimate query q will be received by all sensors in WSN.

4 Basic Authenticated Query Flooding: `AQF-pass`

We now describe our basic protocol for authenticated query flooding, called `AQF-pass`. This protocol uses the *pass* strategy which is explained below in the protocol description. In a nutshell, if the sensor cannot decide whether the query is legitimate or not, it *passes* it to its neighbors. In Section 5 we discuss other possible strategies for this case.

4.1 Preliminaries

ID-Based Key Predistribution Random key predistribution for sensor networks originates from [6]. The idea is that each sensor node is preloaded with randomly chosen k keys, called *key ring*, from the key pool of size l. The values of l and k can be chosen such that any two nodes have at least one common key with a given probability. However, here we do not care about the probability that two neighboring sensors share a key, because in our scheme, key predistribution is used not for secure and authenticated communication between the neighboring sensor nodes, but for authenticated flooding.

ID-based key predistribution was introduced in [22]. The keys in the key pool are numbered from 1 to l. Each sensor s with a unique identifier id_s is first assigned k distinct integers between 1 and l by applying a pseudorandom number generator $PRG()$ with the seed id_s. Then the sensor s is preloaded with the keys whose identifiers are these k numbers from the sequence of pseudorandom numbers $PRG(id_s)$.

This method of choosing key rings enables to characterize sets of key identifiers very efficiently, as only the corresponding short seed needs to be known. This helps to save energy in a sensor network if the set of key identifiers needs to be transmitted over the air, as radio communication is very expensive in terms of energy. In this case, only the seed x is transmitted. Then, any sensor can determine if it knows some keys from a set of key identifiers KID_x characterized by the seed x. It computes $PRG(x) = KID_x$ and compares its own key identifiers to the key identifiers from KID_x.

1-bit MACs In our protocol, we use message authentication codes (MACs) with 1-bit output. The idea of using MACs with single bit output originates from [3]. We view a 1-bit MAC under a given key as a random function, i.e., we require the following:

- A single 1-bit MAC (under an unknown random key) cannot be feasibly guessed with any probability significantly exceeding $\frac{1}{2}$.
- Similarly, an m-bit string of m 1-bit MACs under m independent random keys cannot be guessed with probability significantly more than $\frac{1}{2^m}$.

4.2 AQF-pass: Protocol Description

We assume that an ID-based key predistribution scheme is deployed in the sensor network. The protocol description follows.

Base station The base station first computes the query q and a hash from the query $x = h(q)$ using a hash function $h()$. Then it generates m key identifiers for the underlying ID-based key predistribution scheme: $KID_x = PRG(x) = (kid_1, \ldots, kid_m)$. We denote the corresponding key sequence $K_x = (k_{kid_1}, ..., k_{kid_m})$.

Then the base station computes m 1-bit MACs on $h(q)$ using keys from K_x. We call these m 1-bit MACs *authenticator* for q, denoted as $macs(q)$.

The base station floods the query q into the sensor network, accompanied by the authenticator for the query.

Sensor nodes Upon receiving the query q with the authenticator $macs(q)$, each sensor s first computes $x = h(q)$ and the sequence of key identifiers $KID_x = PRG(x)$. It compares the key identifiers from KID_x to its own key identifiers in order to find out if s knows some keys from K_x.

If s knows some keys, it verifies the corresponding 1-bit MACs from $macs(q)$. If any of them does not verify correctly, then the sensor drops the query. If all verifiable MACs are correct, the sensor forwards the query to his neighbors according to the underlying flooding mechanism.

If the sensor is not able to verify any MACs (i.e., the sensor does not know any keys from K_x), than the sensor forwards the query to his neighbors according to the underlying flooding mechanism. We say that the sensor *passes* the non-verifiable query. This action gives the name to the algorithm `AQF-pass`.

4.3 AQF-pass: Analysis

The query of a legitimate user will be flooded into the sensor network without any obstacles. However, a query forged by an adversary will only be able to reach a limited part of the network, as some sensor nodes will discard the query. In the following, we analytically determine how many 1-bit MACs should be appended to a query in order to limit the propagation of a fake query to a logarithmically small part of the network.

The variables used in the analysis are summarized in Table 1.

Table 1. Variables used in the analysis of AQF-pass.

meaning of the variable	variable	typical values
number of nodes in the sensor network	n	$1000 - 10000$
number of keys in the key pool	l	10000 - 100000
number of keys in the key ring of a node	k	50 - 250
node density (average number of neighbors of a node)	d	5 - 50
number of captured senor nodes	$\tilde{n}$	≤ 30
number of captured keys	$\tilde{b}$	Formula 1
number of keys in the authenticator which the adversary knows	$E_{\tilde{b}}$	Formula 2
number of right bits in the fake authenticator	B	Formula 3
probability that the message will be forwarded	p_f	Formula 4
size of the authenticator	m	100 - 500 bits

Propagation Probability of a Fake Query Using a common model for cryptographic hash functions [2], it is infeasible to first choose some x and then search for an appropriate value q with $h(q) = x$, or to fix *any* properties for the desired x and then search for a query q with satisfying $h(q)$. For different queries q, the adversary always receives independent random values $x = h(q)$.

Therefore, we assume that the adversary uses the following strategy: It computes the seed $x = h(q)$ for its query q, computes the appropriate sequence of key identifiers KID_x using $PRG(x)$, and hopes that it knows enough keys with identifiers from KID_x in order to be able to construct a fake query.

In the following, we compute the probability of a fake query generated as above to successfully propagate through the sensor network assuming that the adversary captured $\tilde{n}$ sensor nodes and guessed the bits of authenticator which it could not compute.

If $\tilde{n}$ sensor nodes are compromised, then the adversary knows in average $\tilde{b}$ keys:

$$\tilde{b} = l(1 - (1 - \frac{k}{l})^{\tilde{n}}) \tag{1}$$

Formula 1 assumes that the keys are distributed according to a uniform probability distribution. Given that the adversary knows $\tilde{b}$ keys out of l, we can compute the average number of bits in a MAC of length m that will be correct due to the adversary's partial knowledge of the key space l:

$$E_{\tilde{b}} = m\frac{\tilde{b}}{l} \tag{2}$$

Since the attacker knows nothing about the other keys in the authenticator, it has to guess the other bits. There it will have the probability of 50% to guess the correct value. This lets us compute the total number of correct bits in the faked authenticator:

$$B = E_{\tilde{b}} + \frac{m - E_{\tilde{b}}}{2} = \frac{m(\tilde{b} + l)}{2l} \tag{3}$$

We can finally compute the probability p_f that a sensor accepts the query with the fake authenticator:

$$p_f = \left(\frac{l-k}{l} + \frac{k}{l}\frac{B}{m}\right)^m \tag{4}$$

The expression in the parentheses gives the probability that one bit of the authenticator passed the test by a particular sensor node. The first summand expresses the probability that the sensor node does not share any keys with the claimed set of key identifiers $PRG(x)$. The second summand shows the probability that either the adversary could compute the appropriate bit or guessed it.

Limiting the Propagation of Fake Queries The last section calculated the probability each single sensors forwards the faked query. It is yet open, however, how the network as a total behaves, namely, whether the query reaches a significant number of nodes, or is stopped from doing so.

To calculate the parameters that have to be set in order to stop a fake query from reaching a critical mass of nodes, we make use of the theory by Erdös and Rényi [5] which is also used in [10]: a random (n,p)-graph[3] becomes disconnected if $pn < 1$, i.e., if the average number of outgoing connections from a node is fewer than 1. In this case the largest connected component is of the size $\Theta(\log(n))$.

In our sensor network, each sensor has d neighbors on average, and each neighbor forwards the query with probability p_f. Then, we have a (d, p_f)-graph for query dissemination and therefore, we have to adjust the parameters that we can control, such that:

$$p_f d < 1 \tag{5}$$

From Formulas 4 and 5 it follows:

$$\frac{1}{d} > p_f \tag{6}$$

$$\Leftrightarrow \frac{1}{d} > \left(\frac{l-k}{l} + \frac{k(\tilde{b}+l)}{2l^2}\right)^m \tag{7}$$

$$\Leftrightarrow m > \frac{\log d}{-\log\left(1 - \frac{k(l-\tilde{b})}{2l^2}\right)} \tag{8}$$

We have variable parameters $d, l, k, \tilde{n}$ (or $\tilde{b}$ instead of $\tilde{n}$) for the length of the authenticator m. The administrators of the network control parameters d, l, k and m, while the adversary controls $\tilde{n}$, and therefore, also $\tilde{b}$.

[3] A random (n,p)-graph has n nodes, and the connection between any two nodes exists with probability p.

The next task is to find suitable ranges for d, l, k and m, such that the adversary is unable to send fake queries for reasonable ranges of $\tilde{n}$. We did so analytically, as well as in a simulation (see Section 4.4).

Reasonably, the more nodes the attacker has compromised, i.e., the more keys it has, the more difficult it gets to keep the attacker from broadcasting illegitimate queries. Also dense networks are harder to protect, due to the high connectivity of query propagation in these networks, as our approach stops an illegitimate query only if the average connectivity of query propagation drops below 1 (Formula 5).

The results of the analytical evaluation using Formula 8 are presented on Figure 1. We considered $n = 1000$ nodes, key pool size $l = 10,000$, node density $d \in \{7, 15\}$ and key ring size $k \in \{75, 150\}$.

Firstly, we comment on the choice of the node density parameter. If the node density is too high, then the capacity of wireless networks decreases. On the other hand, if the network density is too low, the network may become disconnected. Node density required to ensure connectivity can be estimated as $\Theta(\log n)$ [21], but the exact number of neighbors remains an open problem. For networks of moderate size, 6 to 8 neighbors may be considered [17]. Thus, we chose density 15, at which the network should certainly be connected, and density 7 which seems to be a border case.

Figure 1 shows that in the range of tens of compromised nodes, the needed authenticator size increases reasonably slow. The best results are reached for the small node density 7 and the large key ring size 150. In this case, if the adversary captured around 10 sensor nodes, authenticator size around 300 bits suffices to thwart propagation of fake queries.

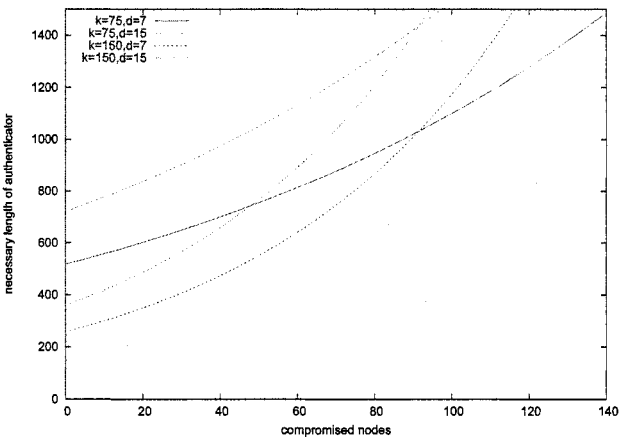

Fig. 1. Necessary authenticator size, depending on node density, key ring size, and the number of compromised nodes. The size of key pool is $l = 10000$.

4.4 Simulation Results

We simulated AQF-pass using Shawn [13], a discrete event simulator for large wireless sensor networks. We used a key pool of $l = 10,000$ keys and varied node density $d \in \{7, 15\}$ and key ring size $k \in \{75, 150\}$. In each simulation run, 1000 nodes were randomly and uniformly placed such that the given node density d was satisfied. The source of the query (base station or the adversary) was also placed randomly in the sensor field. The query was sent, accompanied by the authenticator of size m. We looked into the number of nodes reached by an illegitimate query for $m = 50, 100, 150, \ldots, 500$ assuming that the adversary captured $\tilde{n} = 0, 1, 2, 4, 16, 32, 64, 128$ nodes. For each combination of parameters, 50 protocol runs on different network topologies were performed. Due to space limit, we only present the most significant results here.

At node density 15 and key ring size 75 all the networks were fully connected, that is, legitimate queries always reached all of nodes. However, also illegitimate queries reached a significant part of the network even if no nodes were captured, until the authenticator size reached the unacceptable 500 bits. This confirms our analytical results showing that in this case, authenticator size of around 700 bits are needed.

Formula 8 indicates that the size of the authenticator decreases with the decreasing node density and increasing key ring size. In Figure 2, results for node density 7 and key ring size 150 are presented. With node density 7, the network may already become disconnected. However, the number of sensors which are disconnected from the network in this case was negligible.

With this parameters, authenticator size of 200 bit already suffices for tens of captured nodes as considered in our adversary model. Analytical results suggest authenticator of around 300 bits here.

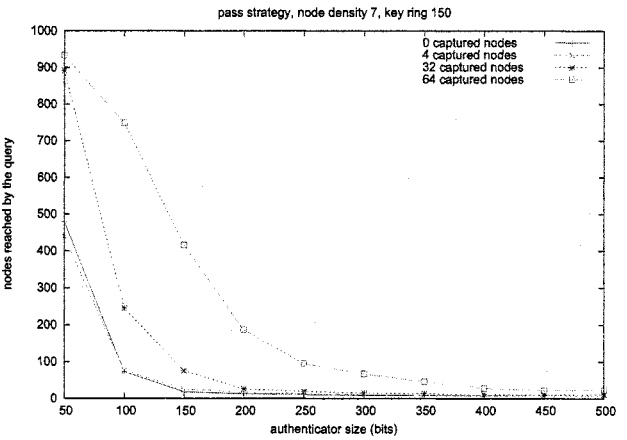

Fig. 2. Number of nodes reached by the fake query depending on authenticator size, with node density 7 and key ring size 150.

4.5 Estimated Verification Efficiency

Although we have not implemented `AQF-pass` yet, we roughly estimate its running time here.

We assume key pool size $l = 10,000$, and key ring size $k = 150$. Therefore, the size of a key identifier is 14 bits, we take 16 bits for convenience of computation. As our analysis and simulations suggest authenticator size m of 200-300 bits, we assume $m = 256$ for convenience. The verification of the query q consists of computing its hash value $h(q)$, generating 256 key identifiers using $PRG(h(q))$, and finally, computing some MACs on $h(q)$. On average, a sensor node would know $\frac{m \cdot k}{l}$ keys from the authenticator. Thus, the sensor node needs to generate $256 \cdot 16 = 4096$ pseudorandom bits, and to compute $\lceil \frac{256 \cdot 150}{10,000} \rceil = 4$ MACs on average.

We assume that both the $PRG()$ and the MAC are implemented using a single block cipher with the block size B as a primitive. As in the following we want to compare the efficiency of `AQF-pass` to existing implementations of other authenticated broadcast protocols, we assume that this cipher is RC5 [19] with $B = 64$.

Pseudorandom numbers can be generated using the block cipher in counter (CTR) mode [15]. Then, generating N pseudorandom bits corresponds to encrypting N bits.

For the MAC computation, we use the CBC-MAC [15], and then take the first bit of its output for the 1-bit MAC. Time to compute CBC-MAC on N bits can be estimated as encrypting B bits $\lceil \frac{N}{B} \rceil$ times. With block size 64, and the output of the hash function 160 bits, computing the CBC-MAC on $h(q)$ can be estimated as 3 encryptions of 64 bit long messages.

At present, the cryptanalysis of hash functions is a very fast developing research area. Therefore, it is difficult to choose a "right" hash function to use. We assume that we use SHA-1 [15]. We very roughly estimate its computation time as time to compute a MAC on 128 bits (the hash function should be more efficient than the MAC). Thus, computing $h(q)$ an be estimated as 2 encryptions of 64 bit long messages.

We compare the verification efficiency of our protocol to the efficiency of two other approaches to authenticated broadcast: μTESLA and digital signatures.

In [8,9], the most efficient (to date) implementation of a well established public key cryptosystem for sensor networks is reported. The authors implemented elliptic curve cryptography (ECC) on the popular MICA2 sensor nodes [4], and obtained time of 0.81s for one point multiplication (key size 160 bits). The digital signatures have length of 320 bits.

On the other hand, [12] reports implementation of RC5 on MICA2 nodes. Encrypting 64 bits with RC5 takes 0.26 ms. Then, computing $h(q)$ takes 0.52 ms, generating 4096 pseudorandom bits takes 16.64 ms, and computing 4 MACs on 160 bits requires $4 \cdot 3 \cdot 0.26 = 3.12$ ms. Thus, query verification in `AQF-pass` requires 20.28 ms, which is much more efficient than digital signatures.

μTESLA [18] is a very efficient authenticated broadcast protocol for sensor networks. It uses only symmetric key cryptography, but its security depends critically on clock synchronization in the sensor network. This incurs additional costs, as even loose clock synchronization in a network with 1000 nodes is a non-trivial task. Verification of μTESLA broadcast messages takes at most 17.8 ms.

μTESLA also uses RC5 as a building block. Encryption of 64 bits in CTR mode takes 0.55 ms, and computing CBC-MAC on 64 bits takes 0.64 ms. Thus, using the same hardware, `AQF-pass` would take 44.16 ms, which is 2.5 times slower than μTESLA. On the other hand, our `AQF-pass` does not require clock synchronization, which may be worth the performance decrease.

5 Discussion and Future Work

Strategies for propagation of non-verifiable queries In the protocol `AQF-pass`, if the node cannot verify the query, it passes the query to all its neighbors. This is only one of possible strategies. Another obvious strategy is `AQF-stop`, where the node drops the non-verifiable query. In this case, the legitimate queries also can be dropped, so care should be taken about their propagation probability as well. We analyzed and simulated this approach, and found out that it does not bring significant improvements. When the authenticator size is too small, legitimate queries do not propagate well, as too many nodes cannot verify the queries and drop them. With the growth of the authenticator size, however, the event *stop* (meaning "a sensor cannot verify the query") gets so rare, that the performance of `AQF-stop` gets very similar to that of `AQF-pass`. Nevertheless, the *stop* strategy seems to be promising, as it decreases the number of sent messages in the network, and therefore, saves energy. Thus, we are looking at how to increase the probability of the drops. We now explore the following strategy: if the sensor could only verify h bits of the authenticator, it drops the query with probability $\frac{1}{2^h}$.

Flooding strategies The protocol `AQF-pass` works efficiently for sparse sensor networks. The reason it gets worse with denser networks lies in the fact that if each sensor node has a lot of neighbors, each forwarding event increases the probability of a query to get trough the network. And as the adversary always can guess at least half of the bits in the authenticator, even queries with completely guessed authenticator propagate successfully. To thwart this disadvantage, we plan to use more sophisticated flooding mechanisms, such as gossiping.

Preventing a sophisticated attack While the proposed protocol has its merits, there is a weakness in the protocol that has to be taken into account. If an attacker is able to send a query q with different authenticator $macs(q)$, it might gain a broader access to the network in case the attacker can observe parts of the network. The attack works as follows: if the attacker has no knowledge about a single bit in $macs(q)$, it sends one message with the bit set, and one with the bit cleared. It can then guess from the number of nodes accepting the message, whether the bit should be set or not. This can be repeated for all bits that are unknown to the attacker, until the message either reaches a sufficient number of sensors, or the attacker knows how to set all bits correctly.

To thwart this attack, it has to be avoided that a message q can be sent with different $macs(q)$. The first proposal to counter the attack is that each sensor stores the number of invalid requests it received from its neighbors. Further messages from a neighbor are only forwarded if the number of invalid queries from it is below a certain threshold,

or according to some probability distribution that depends on this number, e.g. exponentially decreasing probability. This scales quite well, since the amount of data stored is independent from the number of relayed messages, does not need a timestamp, and is equal to the number of a node's neighbors. This counter technique isolates compromised sensors, such that they are unable to send future requests.

Verification efficiency Finally, efficiency of the authenticator verification remains to be determined exactly. We are now working on design of an efficient 1-bit MAC scheme, and plan to implement the query verification on the real sensor nodes.

References

1. Alexander Becher, Zinaida Benenson, and Maximilian Dornseif. Tampering with motes: Real-world physical attacks on wireless sensor networks. In *3rd International Conference on Security in Pervasive Computing (SPC)*, April 2006.
2. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
3. Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Proc. IEEE INFOCOM'99*, volume 2, pages 708–716, New York, NY, March 1999. IEEE.
4. Crossbow, Inc. MICA2 data sheet. Available at `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`.
5. P. Erdös and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.*, pages 17–61, 1960.
6. Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 41–47. ACM Press, 2002.
7. Saurabh Ganeriwal, Srdjan Capkun, Chih-Chieh Han, and Mani B. Srivastava. Secure time synchronization service for sensor networks. In *WiSe '05: Proceedings of the 4th ACM workshop on Wireless security*, pages 97–106, New York, NY, USA, 2005. ACM Press.
8. Vipul Gupta, Matthew Millard, Stephen Fung, Yu Zhu, Nils Gura, Hans Eberle, and Sheueling Chang 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.
9. Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and rsa on 8-bit CPUs. In *Cryptographic Hardware and Embedded Systems (CHES); 6th International Workshop*, pages 119–132, August 2004.
10. Joengmin Hwang and Yongdae Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 43–52. ACM Press, 2004.
11. Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed Diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003.
12. Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004)*, November 2004.

13. A. Kröller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer. Shawn: A new approach to simulating wireless sensor networks. In *Design, Analysis, and Simulation of Distributed Systems, SpringSim 2005*, April 2005.
14. Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 491–502, New York, NY, USA, 2003. ACM Press.
15. Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, 1997.
16. moteiv Corp. Telos revision B datasheet. Available at `http://www.moteiv.com/products/docs/telos-revb-datasheet.pdf`.
17. J. Ni and S. Chandler. Connectivity properties of a random radio network. *IEE Communications*, 141:389–296, August 1994.
18. Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. SPINS: security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002.
19. Ronald L. Rivest. The RC5 encryption algorithm. In *Fast Software Encryption*, pages 86–96, 1994.
20. Stefaan Seys and Bart Preneel. Efficient cooperative signatures: A novel authentication scheme for sensor networks. In *2nd International Conference on Security in Pervasive Computing*, number 3450 in LNCS, pages 86 – 100, April 2005.
21. Feng Xue and P. R. Kumar. The number of neighbors needed for connectivity of wireless networks. *Wirel. Netw.*, 10(2):169–181, 2004.
22. Sencun Zhu, Shouhuai Xu, Sanjeev Setia, and Sushil Jajodia. Establishing pair-wise keys for secure communication in ad hoc networks: A probabilistic approach. In *IEEE International Conference on Network Protocols*, November 2003.