Distributed Key Certification Using Accumulators for Wireless Sensor Networks

Jun-Young Bae[1(✉)], Claude Castelluccia[2], Cédric Lauradoux[2], and Franck Rousseau[1]

[1] Grenoble Institute of Technology, Grenoble Informatics Laboratory – CNRS, UMR 5217, 38402 Saint Martin D'Hères, France
{jun-young.bae,franck.rousseau}@imag.fr
[2] INRIA, 655 Avenue de L'Europe, 38334 St Ismier, France
{claude.castelluccia,cedric.lauradoux}@inria.fr

Abstract. In this work, we propose a key certification protocol for wireless sensor networks that allows nodes to autonomously exchange their public keys and verify their authenticity using one-way accumulators. We examine and compare different accumulator implementations for our protocol on the Sun SPOT platform. We observe that our protocol performs best with accumulators based on Elliptic Curve Cryptography (ECC): ECC-based accumulators have roughly the same speed as Secure Bloom filters, but they have a smaller memory footprint.

Keywords: Wireless sensor networks · One-way accumulators

1 Introduction

Wireless Sensor Networks (WSNs) are envisioned as a key part of what is now known as the *Internet of Things* (IoT). In a few years from now, large sensor networks, composed of myriads of inexpensive low-power and energy-constrained wireless nodes, will be commonplace. However, wireless communications are easy to eavesdrop and manipulate, thus more prone to serious attacks. Securing communications in wireless sensor network is, therefore, of utmost importance.

Two critical issues in WSNs are key distribution and certification. They are mandatory in order to establish confidentiality in the network. Over the last decade, symmetric cryptography was, by far, the most popular primitive used for key distribution and certification since asymmetric cryptography was considered to be computationally too heavy to handle for low-power sensor nodes (*e.g.* [7,9]). However, at the end of the last decade, Elliptic Curve Cryptography (ECC) started to be deployed on sensor nodes [15] which showed that asymmetric cryptography is feasible in WSNs.

If the cost of asymmetric cryptography is now no more an issue, there is still the problem of certifying the public keys of the nodes. We cannot simply preinstall the public key of the Certificate Authority (CA) and the digital certificates that guarantee the authenticity of the public keys of the nodes: any nodes that

I. Stojmenovic et al. (Eds.): MOBIQUITOUS 2013, LNICST 131, pp. 500–511, 2014.
DOI: 10.1007/978-3-319-11569-6_39

are certified by the same CA would potentially be able to communicate with each other, even if they are not supposed to be in the same network (*e.g.* your nodes would start communicating with your neighbor nodes, because they are produced and certified by the same manufacturer). A solution to this problem would be to allow the owner of the network to act as the CA and pre-install within each of the nodes the necessary cryptographic materials. However, this solution would not scale up for large number of nodes. Moreover, if the only way to install the cryptographic materials within the nodes is through the wireless medium, because they are located in places that are inaccessible (*e.g.* embedded inside walls), an attacker may initiate a Man-in-The-Middle (MiTM) attack and swap the public key and the certificates of the owner with its own. The contribution of this paper is to break through this problem by proposing a distributed key certification protocol based on *one-way accumulators* that is fully autonomous and does not require the use of CAs and certificates.

Accumulators are space/time efficient data structures that are used in order to verify if an element belongs to a predefined set. Several accumulator designs are available, ranging from RSA-based accumulators [2] to Secure Bloom filters [11]. We have studied and evaluated these different designs by implementing them on the Sun SPOT platform in order to determine which one is the most practical for our protocol in terms of processing speed, memory footprint, data transmission, and energy efficiency. We observe that our protocol has the best overall performance with ECC-based accumulators.

All the hypothesis used throughout this paper, for the nodes and the adversary, are defined in Sect. 2. The accumulators, as well as our protocol for distributed key certification, are described in Sect. 3. A brief security analysis of our protocol is available in Sect. 4. The performance evaluation and the impact of the choice of accumulators are discussed in Sect. 5. Previous work in the existing literature are reviewed in Sect. 6. Finally, Sect. 7 concludes our work and proposes future research directions.

2 Assumptions

In this work, we focus on securing large multi-hop networks that may be deployed alongside other unrelated networks. Wireless sensor nodes are low-power embedded devices. A gateway is used in order to collect data and/or relay traffic from the nodes that may be placed at any arbitrary position and may even be mobile. We need to be able to distinguish between different legitimate networks in order to prevent unwanted information leakage or unwillingly transiting external traffic (*i.e.* waste energy). We also need to prevent attackers from joining or tampering with the network.

2.1 Node Model

We assume that the nodes are *interface-less*: no physical interaction with the node is possible and the only way to communicate with them is through the

wireless medium. This assumption is realistic considering the cases of *intelligent buildings* and *IoT*, in which nodes might be, for example, embedded inside walls. Manufacturers also tend to integrate various sensors into a single chip (System-on-Chip) so that they are easier to miniaturize, mass-produce, and deploy in a non-invasive way. We assume that these nodes are produced and distributed in batches by the manufacturer. This is a reasonable assumption, considering that manufacturers would probably not sell cheap, mass-produced nodes individually, and that at least a few nodes are needed in order to build a practical multi-hop WSN. We also consider that customizing nodes with specific cryptographic materials at production time is not much of a limitation. This process is already quite common for certain chip manufacturing, in which unique identifiers or calibration data are loaded after fabrication.

2.2 Attacker Model

Our solution is designed to deal with three particular threats: *Node injection*, *Node capture*, and *Denial-of-Service*.

Node injection – The adversary must not be able to inject its own node in the network.

Node capture – The adversary can gain full control of a node since we assume that the nodes are not *tamper-resistant*. This implies that all the node secrets are exposed to the adversary. In order to measure the resistance of a protocol, the number of communications that can be eavesdropped by the adversary after the capture of a node, is considered (see [8]). It does not include the communications of the compromised node.

Denial-of-Service (DoS) – The adversary may attempt to exhaust the resources of the nodes by sending bogus messages or requests. The protocol must protect all the computationally intensive operations: the adversary must not be able to trigger them easily.

We have considered that two attacks were outside the scope of this paper. A *Relay attack* is the most basic form of a MiTM attack. They can be used in order to mount more sophisticated attacks such as *Wormhole attacks* or DoS. Various solutions to this problem have been proposed [12,20]. In *Replication attacks*, the adversary injects nodes in the network that are copies of a legitimate node. Specific solutions to this problem exist [18].

3 Accumulator-Based Protocol

After reviewing the principles of different one-way accumulators, we describe how to choose and use them in order to verify keys in the context of sensors networks.

3.1 One-Way Accumulator

One-way accumulators are authenticated data structures similar to Bloom filters [3]. They were introduced by Benaloh and de Mare [2] under the name of

Accumulated Hashing. The purpose of an accumulator is to allow a user to decide whether an item belongs to a given set or not. This probabilistic data structure is known to be efficient in time and space at the cost of a probability of false positives.

Let us define two sets $\mathcal{A}$ and $\mathcal{B}$. $\mathcal{A}$ is the set of the accumulator and $\mathcal{B}$ is the set of the items to be accumulated. An accumulator for set $\mathcal{S} \subset \mathcal{B}$ of i items v_i is denoted $\mathbf{z}^i$. Accumulators are based on commutative functions.

Definition 1. *Function $F : \mathcal{A} \times \mathcal{B} \rightarrow \mathcal{A}$ is said to be commutative (or quasi-commutative [2]) if: $F(F(a,b),c) = F(F(a,c),b), \forall a \in \mathcal{A}$ and $b, c \in \mathcal{B}$.*

All accumulators implement the following components:

Key-Gen(s, K_{priv}, f) – Given security parameter s, master key K_{priv}, and key derivation algorithm f, it generates all the required key materials.

Build-Acc$(\mathbf{z}^0, v_1, \cdots, v_m)$ – From a commutative function, seed $\mathbf{z}^0 \in \mathcal{A}$ and m items $v_i \in \mathcal{B}$, value $\mathbf{z}^m$ is computed recursively:

$$\mathbf{z}^i = F(\mathbf{z}^{i-1}, v_i), \ i \ \text{ in } \ 1, \cdots, m.$$

Gen-Wit – Membership witness w_i is a value associated with each value v_i accumulated in $\mathbf{z}$. The witness is used during the verification of v_i.

Authenticate$(\mathbf{z}^m, v_i)$ – It decides whether item v_i belongs to accumulator $\mathbf{z}^m$. When a witness is needed, Authenticate verifies if the following equality is satisfied: $\mathbf{z}^m \stackrel{?}{=} F(w_i, v_i)$.

For simplicity, we use the notation $\mathbf{z}$ throughout the paper instead of $\mathbf{z}^i$.

3.2 Choosing an Accumulator

In this section, we present the available implementation choices for an accumulator based on asymmetric or symmetric cryptography.

Asymmetric Accumulators. In the original paper [2], Benaloh and de Mare suggested to use the modular exponentiation as a quasi-commutative function in order to create a one-way accumulator: $F(a,b) = a^b \bmod n$, with $a \in \mathcal{A}$ and $b \in \mathcal{B}$. For an appropriate choice of n, this function is one-way as long as the RSA assumption [19] holds. The Benaloh and de Mare accumulator works as follows:

Gen – This function generates all the values required by function F. Let us define $n = pq$ as the product of two safe primes p, q of approximately the same size. Further details on the security issues concerning the choice of n can be found in the original paper. We only review its main ideas in this section.

Build-Acc – Seed $\mathbf{z}^0$ is also needed in order to bootstrap the accumulator. Therefore, value $\mathbf{z}^i$ is computed recursively: $\mathbf{z}^i = F(\mathbf{z}^{i-1}, v_i)$.

Gen-Wit – With each value v_i belonging to accumulator $\mathbf{z}$, we associate *witness* denoted w_i. Witness w_i corresponding to item v_i is a *partial accumulator* that includes all the values in $\mathbf{z}$ except v_i.

Authenticate – Value v_i that belongs to $\mathbf{z}$ verifies the following equation:

$$F(w_i, v_i) = w_i^{v_i} \bmod n = \left(\mathbf{z}^0\right)^{\prod_{j=1}^{m} v_j} \bmod n = \mathbf{z}.$$

Other quasi-commutative functions have been proposed [1,26]. The scalar-point product over elliptic curves is a natural choice. Let us denote a as a scalar and P, Q as two points of an elliptic curve such that: $Q = aP$. The Elliptic Curve Discrete Logarithmic Problem guarantees that it is computationally infeasible to deduce a from the knowledge of P and Q [13]. Therefore, Authenticate function for an ECC-based accumulator $\mathbf{z}$, item v_i, and witness w_i, is:

$$F(w_i, v_i) = v_i w_i = \left(\prod_{j=1}^{m} v_j\right) \mathbf{z}^0 = \mathbf{z}.$$

Symmetric Accumulators. A symmetric-key accumulator equivalent to the Benaloh and de Mare accumulators was proposed by Nyberg [17]. It is based on cryptographic hash functions and has a much simpler Authenticate function, because it does not require partial accumulators. The connection between Nyberg accumulators and Bloom filters was later made by Yum *et al.* [25] who also demonstrated that Bloom filters are better cryptographic accumulators than Nyberg accumulators in terms of the minimal false positive rate. In parallel and independently of the Nyberg's results, Goh proposed *Cryptographic/Secure Bloom filters* [11] with applications to private information retrieval in databases.

A symmetric accumulator is considered here as a ℓ-bit vector, $\mathbf{z} = (\mathbf{z}_1, \cdots, \mathbf{z}_\ell)$ with $\mathbf{z}_i \in \mathbb{F}_2$. The support of an ℓ-bit vector, denoted as $\mathrm{supp}(z)$, is the set of the non-zero coordinate indexes: $\mathrm{supp}(z) = \{i \in [1, \ell], z_i \neq 0\}$.

Build-Acc$(\mathbf{z}^0, v_1, \cdots, v_m)$ – The creation of the accumulator is done for m items (keys) using the following recursion: $\mathbf{z}^i = \mathbf{z}^{i-1} \vee g(v_i), i \;\text{ in }\; 1, \cdots, m$ with $\vee$ the bitwise inclusive-or operator and g is a function from $\mathcal{B}$ to $\mathcal{A}$.

Authenticate$(\mathbf{z}, v_i)$ – We can observe from the previous equation that:

$$\mathrm{supp}(\mathbf{z}) = \mathrm{supp}(\mathbf{z}^{m-1}) \cup \mathrm{supp}(g(v_i)).$$

Therefore, item $v_i \in \mathcal{B}$ belongs to $\mathbf{z}$ if: $\mathrm{supp}(g(v_i)) \subset \mathrm{supp}(\mathbf{z})$.

For Bloom filters, Key-Gen function must generate k secrets keys, K_i from secret s. If H is a cryptographic hash function that can be keyed, function g is defined by:

$$\mathrm{supp}(g(v_i)) = \{H(K_i, v_i),\ i \in 1, \cdots, k\}.$$

The differences between Bloom filters and Secure Bloom filters are: (a) the use of HMAC-SHA-1 as a hash function and (b) the use of implementation trade-offs in order to reduce the number of computed hashes [14]. It is worth mentioning that many libraries use cryptographic hash functions for Bloom filters instead of universal hash functions.

False Positive Probability. Since accumulators are probabilistic data structures, they have to be carefully designed in order to control their false positive probability. Bari and Pfitzmann have shown how to reduce the false positive problem to the strong RSA assumption under certain conditions [1]. Benaloh and de Mare have shown that the false positive probability is negligible for $|n| \geq 1024$ bits. The same can be said about accumulators based on the 160-bit prime field ECC. For Bloom filters, the false positive probability p is shown to be: $\mathsf{p} = \left(1 - \left[1 - \frac{1}{\ell}\right]^{km}\right)^k$ [3].

3.3 Protocol

The protocol described in this paper is based on three functions: **initialization**, **ownership transfer**, and **node-to-node key verification**.

Initialization – The manufacturer assigns to each node N_i a pair of public/private keys denoted as (PK_{N_i}, SK_{N_i}). The public keys of all nodes PK_{N_i} belonging to the network are accumulated in $\mathbf{z}$. In addition, the manufacturer generates a public/private key pair for the gateway and includes the public key of the gateway in accumulator $\mathbf{z}$ (*i.e.* the gateway is considered as any other node). The manufacturer stores $\mathbf{z}$ in every node. In addition, every node is set up with all the values needed for the accumulator including the witness w_i.

Ownership transfer – The final user of the network uses a gateway in order to manage and monitor the network. In order to transfer the ownership, the manufacturer provides the public/private key pair of the gateway to the final user. We assume that at a given time there is only one owner. The manufacturer needs to address two issues before transferring the ownership to the final user: (a) the gateway must be able to distinguish the nodes that belong to the network and (b) the nodes must recognize the public-key of the gateway. The first issue is solved by transferring $\mathbf{z}$ and all the necessary parameters from the manufacturer to the gateway. The gateway can then execute the $\mathsf{Authenticate}$ function just like any other node. The second problem is solved by including the public key of the gateway in $\mathbf{z}$.

Node-to-node key verification – As shown in Fig. 1, the nodes exchange their public-keys PK_{N_i} with their corresponding witnesses w_i. They then execute the Verify function described in Algorithm 1. When a node determines that a public key belongs to the accumulator, it stores it as an entry in table T. The $\mathsf{Lookup}(T, PK_{N_i})$ function verifies if the node has already verified the public key of node N_i. The $\mathsf{Put}(T, PK_{N_i})$ function adds the public key of the node to the table. Once the nodes have mutually verified their public keys with Verify, they can establish a symmetric key through the Elliptic Curve Diffie-Hellman (ECDH) key agreement protocol [16].

4 Security Analysis

Our protocol inherits the major security properties of one-way accumulators: "one-way-ness" and resistance to forgery [1,2].

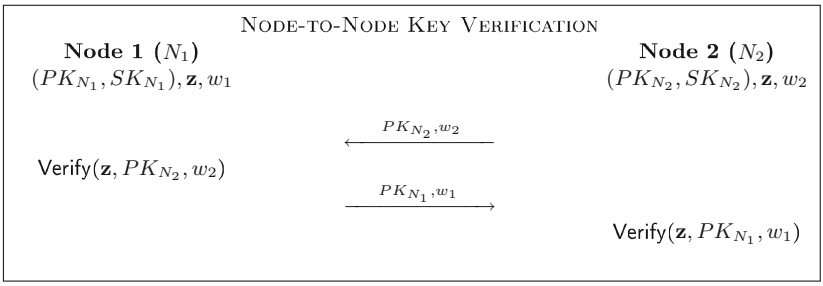

Fig. 1. Key certification protocol based on accumulators.

Algorithm 1. Verify($\mathbf{z}, PK_{N_1}, w_1$) function ($N_1$ by N_2).

```
if Lookup(T,PK_{N_1})= false then
  if Authenticate(z, PK_{N_1}, w_1)= true then
    Put(T, PK_{N_1})
  end if
else
  Do nothing (key already verified)
end if
```

Node injection – The first attempt of the adversary might be to send its own public key and witness. In this case, the security of our protocol is reduced to the security of the accumulator.

Node capture – Due to the "one-way-ness" of accumulators, the capture of a node does not compromise the communications of other nodes: only the keys of the captured node are compromised.

Denial-of-Service (DoS) – The goal of the adversary is to make the nodes waste precious resources (*e.g.* energy). In order to achieve this goal, computationally expensive operations are triggered. We assume that the most expensive operation is ECDH and our protocol prevents an adversary to trigger useless ECDH computations. The lookup table T prevents the replay of correct messages that cause the exhaustion of node resources.

Please note that since the manufacturer produces each batch of nodes with distinct, pre-installed accumulators, the problem of "promiscuous" connections between certified nodes that should not belong to the same network, as well as the MiTM attack during the wireless pre-installation of cryptographic material that were mentioned in Sect. 1 become irrelevant.

5 Performance Evaluation

We have implemented the one-way accumulators of Benaloh and de Mare (we will call them RSA and ECC-based accumulators) and the Secure Bloom filter

(hereafter simply referred to as Bloom filter) in Java ME[1] on Sun SPOTs[2]. Each Sun SPOT comes with a 180 MHz 32-bit ARM920T processor, 512 kB RAM, 4 MB Flash memory and a 3.7 V rechargeable 750 mAh lithium-ion battery. They have an integrated TI CC2420 radio chip operating in the 2.4 GHz band and compliant with IEEE 802.15.4. They also come with a readily available security API[3]. The elements accumulated in our implementation are 160-bit Elliptic Curve (EC) public keys using the *secp*160*r*1 curve domain parameters[4]. We have used the SHA-1 hashing algorithm[5] that generates 160-bit hash codes.

Once we have implemented all the different accumulators, we have measured the time that the various computations and transmissions within our protocol take in order to evaluate the total time required for a node-to-node key verification on Sun SPOTs. The entire node-to-node key verification between two Sun SPOTs (let us call them SPOT 1 and SPOT 2) using asymmetric accumulators (RSA-based and ECC-based accumulators), can be divided into four steps:

1. SPOT 1 sends its public key PK_1 and its witness w_1 to SPOT 2.
2. SPOT 2 executes $\mathsf{Verify}(\mathbf{z}, PK_1, w_1)$.
3. SPOT 2 sends its public key PK_2 and its witness w_2 to SPOT 1.
4. SPOT 1 executes $\mathsf{Verify}(\mathbf{z}, PK_2, w_2)$.

Node-to-node key verification using Bloom filters would only differ by the lack of witnesses in each step. We can see that step 3 to 4 is simply a repetition of step 1 to 2. Therefore, we have implemented step 1 and 2 to run between two SPOTs and we have then measured the duration of each step. Table 1 outlines the values of the parameters used for the different accumulator implementations. These values ensure an equivalent level of security for each implementation. Table 2a shows the results for each step with the implementations based on asymmetric accumulators and Bloom filters. For each measurement, one thousand 160-bit EC public keys have been accumulated. The time necessary for a node-to-node key verification between two SPOTs is obtained by multiplying the total time by two (*e.g.* $1.4 \times 2 = 2.8$ s for the key verification using Bloom filters).

The time that step 2 takes mostly consists of the execution time of the $\mathsf{Authenticate}$ function in Verify (see Sect. 3.3). We should also note that, although the time that step 2 takes when using Bloom filters is slightly less than when using ECC-based accumulators, it becomes equivalent or longer once p is set to 2^{-90} or less.

We have also estimated the energy consumption on the Sun SPOT based on the duration of each steps of the node-to-node key verification (*cf.* Table 2b). The most "energy-efficient" key verifications are the ones using Bloom filters and ECC-based accumulators. The extra energy that the key verification using ECC-based accumulators requires from step 1 to 3 compared with the one using

[1] http://www.oracle.com/technetwork/java/javame/
[2] http://www.sunspotworld.com/
[3] http://java.net/projects/spots-security
[4] http://www.secg.org/collateral/sec2_final.pdf
[5] http://www.ietf.org/rfc/rfc3174.txt

Table 1. Parameter values used in the accumulators.

Accumulator	Parameter	Value
RSA-based	Length of n	1024 bit
ECC-based	Curve domain parameters	$secp160r1$
Bloom	Value of p	2^{-80}

Table 2. Duration and energy consumption of node-to-node key verification.

(a) Duration of the first two steps in node-to-node key verification.

	Time (s)		
Step	**RSA**	**ECC**	**Bloom**
1	1.1	1.1	1.0
2	4.4	0.5	0.4
TOTAL	**5.5**	**1.6**	**1.4**

(b) Energy consumption during node-to-node key verification.

	Consumption (mAh)		
Step	**RSA**	**ECC**	**Bloom**
1	0.029	0.028	0.027
2	0.033	0.007	0.006
3	0.015	0.013	0.013
4	0.099	0.011	0.009
TOTAL	**0.176**	**0.059**	**0.055**

Bloom filters is due to the additional exchange of witnesses. However, the amount of energy consumed in step 4 of the key verification using Bloom filters would grow as the execution time of the Verify function increases due to a lower p.

Table 3a outlines the size of the public/private keys, accumulators and witnesses, used in node-to-node key verification. We can notice that much less memory is required in order to store asymmetric accumulators than Bloom filters. The advantage of using an ECC-based accumulator is apparent here, since it is only $(48 + 48)/14427 \approx 1/150$ of the size of a Bloom filter. Additionally, the

Table 3. Memory footprint of node-to-node key verification.

(a) Size of the variables used in node-to-node key verification.

	Size (bytes)		
Instance	**RSA**	**ECC**	**Bloom**
PK	1338	1338	1338
SK	1261	1261	1261
z	128	48	14427
w	128	48	N/A
TOTAL	**2855**	**2695**	**17026**

(b) Memory usage in node-to-node key verification.

	Usage (kB)		
Step	**RSA**	**ECC**	**Bloom**
1	79	78	78
2	0.0	0.0	0.0
3	53	43	36
4	3.1	3.6	1.6
TOTAL	**135**	**125**	**116**

size of Bloom filters rapidly increases as more elements are accumulated since they need to increase the number of bits per element in order to maintain their false positive probability [3]. Since the size of asymmetric accumulators is constant regardless of the number of accumulated elements, we can conclude that asymmetric accumulators have much better scalability in terms of accumulator size compared to Bloom filters. From the memory usage measurements shown in Table 3b, we can see that the two most "memory-efficient" key verifications are the ones using Bloom filters and ECC-based accumulators. However, the memory required in step 4 of the key verification using Bloom filters would increase if the number of accumulated keys increases or if p is set lower.

6 Related Work

Key establishment and distribution are critical problems in WSNs that have been deeply investigated by the community. Two schools are in competition on this topic: symmetric cryptography and asymmetric cryptography. Our work naturally belongs to the latter. Due to the lack of space, we will only briefly review the work based on asymmetric cryptography. We encourage the readers who are interested in further details on the work based on symmetric cryptography to consult [4,8] for a complete survey.

Asymmetric cryptography (more specifically, RSA-based cryptosystems) was, for a while, considered infeasible for sensor nodes. TinyPK [24] was the first attempt to implement a public-key infrastructure in WSNs. It uses the classical Diffie-Hellman (DH) protocol and a CA in order to certify the keys. The performance results were relatively modest and not promising for a practical application. Since then, TinyECC [15] and NanoECC [22] libraries proved the feasibility and the efficiency of ECC on sensor nodes. Low-cost hardware extensions have also been demonstrated [10,23]. The ECDH protocol has also been put into practice on WSNs. For example, Sun *et al.* used a "self-certified" ECDH protocol along with a polynomial-based weak authentication scheme in order to thwart DoS attacks initiated by bogus ECDH requests [21]. Our approach is the natural extension of these work: accumulators eliminate the need for CAs and certificates during key establishment.

7 Conclusion and Future Work

In this paper, we have proposed a key certification protocol that does not require CAs and certificates, in which nodes can autonomously verify the authenticity of the public keys they exchange using one-way accumulators. Our results show that the exchange and verification of public keys between two wireless sensor nodes can be performed within few seconds, while consuming little energy. Our analysis of the existing accumulators lets us conclude that ECC-based accumulators are best suited for our protocol since they perform as fast as Bloom filters while having a memory footprint that does not increase with the number of accumulated elements.

Our next immediate objective is to study how our protocol can be adapted for current WSN standards (802.15.4[6], RPL[7] etc.). Although many nodes produced nowadays are as powerful as Sun SPOTs (*e.g.* STM32W[8], albeit without a Java Virtual Machine), our protocol should also be tested on nodes with less processing power. Another issue that needs to be addressed is that our protocol cannot safely add and remove nodes from the network. Using dynamic accumulators [6] that allow you to add and remove elements may be a solution to this problem, but security flaws have been found in these designs [5].

Acknowledgement. This work was partially supported by the French National Research Agency projects ARESA2 and IRIS under contracts ANR-09-VERS-017 and ANR-11-INFR-016 respectively, and the European Commission FP7 project CALIPSO under contract 288879.

References

1. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997)
2. Benaloh, J.C., de Mare, M.: One-way accumulators: a decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
3. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**, 422–426 (1970)
4. Buttyan, L., Hubaux, J.-P.: Security and Cooperation in Wireless Networks. Cambridge University Press, Cambridge (2007)
5. Camacho, P., Hevia, A.: On the impossibility of batch update for cryptographic accumulators. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 178–188. Springer, Heidelberg (2010)
6. Camenisch, J.L., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 61. Springer, Heidelberg (2002)
7. Chan, H., Perrig, A.: PIKE: peer intermediaries for key establishment in sensor networks. In: INFOCOM, March 2005, pp. 524–535. IEEE (2005)
8. Chan, H., Perrig, A., Song, D.: Key distribution techniques for sensor networks. In: Raghavendra, C.S., Sivalingam, K.M., Znati, T. (eds.) Wireless Sensor Networks, pp. 277–303. Springer, New York (2004)
9. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: ACM Conference on Computer and Communications Security - CCS 2002, November 2002, pp. 41–47. ACM (2002)
10. Fan, J., Batina, L., Verbauwhede, I.: HECC goes embedded: an area-efficient implementation of HECC. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 387–400. Springer, Heidelberg (2009)

[6] http://www.ieee802.org/15/pub/TG4.html
[7] http://www.ietf.org/rfc/rfc6550.txt
[8] http://www.st.com/web/en/catalog/mmc/FM141/SC1169/SS1581

11. Goh, E.-J.: Secure indexes. Cryptology ePrint Archive, Report 2003/216. http://eprint.iacr.org/2003/216/ (2003)
12. Gollakota, S., Ahmed, N., Zeldovich, N., Katabi, D.: Secure In-Band wireless pairing. In: USENIX Security Symposium, August 2011
13. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer, New York (2004)
14. Kirsch, A., Mitzenmacher, M.: Less hashing, same performance: building a better Bloom filter. Random Struct. Algorithms **33**(2), 187–218 (2008)
15. Liu, A., Ning, P.: TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks. In: International Conference on Information Processing in Sensor Networks - IPSN 2008, April 2008
16. NIST National Institute of Standards and Technology. Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised). NIST Special Publication 800-56A, March 2007
17. Nyberg, K.: Fast accumulated hashing. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 83–87. Springer, Heidelberg (1996)
18. Parno, B., Perrig, A., Gligor, V.D.: Distributed detection of node replication attacks in sensor networks. In IEEE Symposium on Security and Privacy - S&P 2005, May 2005
19. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**, 120–126 (1978)
20. Singelée, D., Preneel, B.: Key establishment using secure distance bounding protocols. In: International Conference on Mobile and Ubiquitous Systems - MobiQuitous 2007, August 2007
21. Sun, K., Liu, A., Xu, R., Ning, P., Maughan, W.D.: Securing network access in wireless sensor networks. In: ACM Conference on Wireless Network Security - WISEC 2009, March 2009
22. Szczechowiak, P., Oliveira, L.B., Scott, M., Collier, M., Dahab, R.: NanoECC: testing the limits of elliptic curve cryptography in sensor networks. In: Verdone, R. (ed.) EWSN 2008. LNCS, vol. 4913, pp. 305–320. Springer, Heidelberg (2008)
23. Verbauwhede, I.: Low budget cryptography to enable wireless security. In: ACM Conference on Wireless Network Security, Invited talk, June 2011
24. Watro, R.J., Kong, D., fen Cuti, S., Gardiner, C., Lynn, C., Kruus, P.: TinyPK: securing sensor networks with public key technology. In: ACM Workshop on Security of Ad Hoc and Sensor Networks - SASN 2004, October 2004
25. Yum, D.H., Seo, J.W., Lee, P.J.: Generalized combinatoric accumulator. IEICE Trans. Inf. Syst. **E91.D**(5), 1489–1491 (2008)
26. Zachary, J.: A decentralized approach to secure management of nodes in distributed sensor networks. In: IEEE Military Communications Conference - MILCOM '03, October 2003