Introduction to Laboratory Research : Evaluation of a Distributed Key Certification using Accumulators in very Constrained Wireless Sensor Networks

André Naz[1]
Supervisors: Jun-Young Bae[2], Franck Rousseau[2]

Grenoble Institute of Technology
ENSIMAG
38402 Saint Martin d'Hères – France
surname.name@ensimag.imag.fr

Grenoble Institute of Technology
Grenoble Informatics Laboratory – CNRS, UMR 5217
38402 Saint Martin d'Hères – France
surname.name@imag.fr

Abstract

In this work we demonstrate that the distributed key certification accumulator based protocol proposed in [1] works well on low-power and memory constrained devices such as WSN430 and MB851 sensors. We observed that in large-scale sensor networks, accumulators based on elliptic curve cryptography (ECC) are much more convenient to use than secure Bloom filters because of their larger memory footprint. Obtained times of computation are very promising, and totally acceptable. In small-scale networks, computations using Bloom filters are faster than with ECC-based accumulators.

1 Introduction

This work is part of the projects Calipso and Iris on wireless sensor networks in which is involved the Drakkar group of the Computer Lab of Grenoble (LIG) in partnership with other European laboratories.

In the foreseeable future large sensor networks composed of thousands low-power, memory and energy constrained sensors will be commonplace [1]. Although wireless communications are easy to eavesdrop, security in such networks has long been set aside. Since confidential data could be transferred, it is without doubt a key issue. Over the last decade symmetric cryptography was used in favor of asymmetric cryptography because of a major problem in distributed networking: key distribution and certification [1]. Using certification authorities like in the Internet is not conceivable. Indeed nodes have to be authenticated before accessing the network. Moreover those authorities can not be always reachable and communications increase network traffic and energy consumption [1].

[1] and [2] recently proposed a decentralized key certification scheme using accumulators in order to solve this problem. However network has to be static from the deployment phase. Nodes can not be added to it. [1] gives performance evaluation of this protocol with different accumulators among symmetric and asymmetric accumulators on Sun SPOT sensor networks. The most efficient accumulator in each family respectively appears to be the secure Bloom filter and the elliptic curve cryptography (ECC) based accumulator [1].

However these kinds of nodes are much more powerful than common basic sensors (T-Mote Sky: CPU: MSP430 16 bit 8MHz, memory: 10KB RAM - 48KB Flash; Sun SPOT v6 : CPU: ARM920T core 32 bit 400 MHz, memory: 1MB RAM - 8MB Flash [3], [4]). The contribution of this paper is to benchmark this protocol in terms of time of execution and memory usage on very constrained devices. Indeed storage, time occupancy and thus energy consumption are major problems when programming on such kinds of sensors. We used ECC-based accumulators and secure Bloom filters. Observed times of computation are perfectly acceptable. In small networks, authentication using a Bloom filter is faster. However, in large-scale networks, the ECC-based accumulator is much more convenient to use, and a Bloom filter is mostly impossible to store on low-memory sensors.

This paper is organized in 8 parts. Section 2 describes the elliptic curve cryptography principle and Section 3 the cryptographic accumulators. A simplified version of the protocol for distributed key certification is presented in Section 4. The benchmarking environment is then shown in Section 5. The Performance evaluation is given in Section 6. A similar work is presented in Section 7. Finally Section 8 concludes our work.

2 Elliptic curve cryptography

ECC was proposed independently by Koblitz [5] and Miller [6] as an approach to public-key cryptography.

In public-key systems everyone has a pair of keys composed of a public key and a secret key. The public key used to encrypt the message is known by everyone. However this key does not allow to decrypt the message. The recipient uses the secret key, which is only known by him, to decipher the message.

In ECC the sender and the receiver publicly agree on a curve E and a point of the curve G. Recommended elliptic curve domain parameters had been published in [7]. Each party generates its key pair, composed of its private key d, which is a random scalar, and its public key Q, which is a point of E computed by $Q = dG$.

Security is based on the discrete elliptic curve logarithm problem, which assures that it is infeasible to find d knowing G and dG in a polynomial time [5]. Elliptic curve cryptosystems are considered as more secure because this problem is much more complex than the classical discrete logarithm [5]. Thus keys can be smaller and still ensure a satisfying level of security. For example 160-bit ECC is as stronger as 1024-bit RSA [7].

3 Cryptographic accumulators

Cryptographic accumulator principle was introduced in [8] by Benaloh *et al.*. Accumulators are probabilistic data structures providing an efficient way to verify if an element belongs to a set. As a consequence, they represent a decentralized alternative to digital signatures and thus eliminate the need of a trusted authority. Accumulators admit a false positive rate. However there are no false negatives. Accumulators are based on one-way and commutative functions [8].

Definition 1. *[8] A function $f : X \times Y \rightarrow X$ is said to be quasi-commutative if :*

$$\forall x \in X, \forall y_1, y_2 \in Y,\ f(f(x, y_1), y_2) = f(f(x, y_2), y_1)$$

For the moment there is no way to remove an accumulated key from an accumulator.

We distinguish two types of accumulators, asymmetric ones and symmetric ones.

3.1 Asymmetric accumulators

Let us suppose a group of n sensors, a one-way, quasi-commutative function $h : X \times Y \to X$ and $x \in X$, the initial value of the accumulator. Each node N_i has a public key $PK_{N_i} \in Y$ and a secret key SK_{N_i}. Every public key can be accumulated in $z \in X$ by:

$$z = h(h(...h(h(x, PK_{N_1}), PK_{N_2}), ...PK_{N_{n-1}}), PK_{N_n}).$$

Since h is a quasi-commutative function, order of accumulation is not important. Each member saves z and w_{N_i}, which represents the accumulated hash of all PK_{N_j} except for $j = i$. Verifying $z = h(w_{N_i}, PK_{N_i})$ authenticates the membership to the network of the node N_i.

Originally Benaloh *et al.* suggested to use the modular exponentiation as a quasi-commutative function [8].

[2] and [1] introduced ECC-based accumulators. It uses the scalar-point product over elliptic curves as one-way, quasi-commutative function [1]. PK_{N_i}, w_i and z are points of the curve. In order to compute the accumulator z from a partial accumulator z_i, a public key is firstly transformed in a scalar which is then multiplied by z_i. False positive rate is negligeable for accumulators based on curve parameter sets from 160-bits and onward [1]. Size of the accumulator and time of computation depend on the used curve.

3.2 Symmetric accumulators

A symmetric accumulator is a l-bit vector.[9] demonstrated that Bloom filters [10] can be used as cryptographic accumulators and even excel other symmetric accumulators. Thus we will only present how secure Bloom filters work. Consider a group of n sensors, a one-way function $g : Y \to X$, and $x \in X$ the initial value of the accumulator, which is just a vector of 0. Each node N_i has a public key $PK_{N_i} \in Y$ and a secret key SK_{N_i}. Every public key can be accumulated in $z \in X$ by:

$$z = x \vee g(PK_{N_1}) \vee g(PK_{N_2}) \vee ... \vee g(PK_{N_n})$$

where $\vee$ is the bitwise inclusive or.

Each member saves z, the l-bit vector. Authenticating membership of a node N_i consists in computing $a = g(PK_{N_i})$ and verifying that $\forall k \in 0..l$ $a_k = 1$ implies $z_k = 1$.

Membership testing is easier with symmetric accumulators since it does not require any partial accumulator.

In secure Bloom filters, g is based on HMAC-SHA-1 function [1]. The size of the accumulator, l, and as a consequence the time of computation increase with the number of elements in the filter or if the false positive rate is set lower.

4 Simplified protocol

For our experiments, the protocol described in [1] can be simplified in two steps. Firstly, the network administrator assigns to each node its public key and its private key. All these public keys are then accumulated. This accumulator is stored in every node. Additionally, every node owns its partial accumulator when an asymmetric accumulator is used. During the node-to-node key verification, nodes exchange their public keys (and possibly their partial accumulators) and mutually verify that the received public keys belong to the accumulator. Involved sensors can then establish a connection through known protocols such as the Elliptic Curve Diffie-Hellman [11].

Here the network is static, since we can not add any other node once every node has been configured. Moreover it is impossible to remove an accumulated key.

5 Benchmarking environment

5.1 Platform

We have implemented ECC-based accumulators and Bloom filters in C. these applications run with Contiki OS [12], an open-source operating system written in C, especially designed for low-power and memory-constrained systems. Contiki is highly portable and runs on a dozen of different systems. It is a non-preemptive OS based on protothreads [13], meaning that a task is run until its completion. Thus, performance are really important: while computing the membership of a node to the network, a sensor will not do its job. Moreover if computation time is too long, the sensor will probably miss some deadlines. Measurements have been done on Conitki OS 2.5.

We experimented our programs on SensLAB [14], a large WSN430 platform, and in our lab on MB851 nodes. Table 1 shows the features of these sensors. Measurements on SensLAB sensors have been done with Contiki 2.5 for SensLAB which includes additional radio drivers.

Table 1: WSN430 and MB851 features

	WSN430	MB851
Micro-controler	MSP430F1611 8 MHz, 16 bits 10KB RAM 48KB flash	STM32W108CB 24 MHz, 32 bits 8KB RAM 128KB flash
External memory	1MB flash	/

In order to implement the ECC-based accumulator, we used Contikiecc library [15] which provides an ECC API. It has been included in Contiki IPsec and IKEv2 [1] [16]. Our Bloom filter implementation is a partial adaptation of the desktop implementation of Erik Nordström to embedded systems [2].

In ECC-based accumulators, transforming a public key in a scalar has been done by hashing the sum of the hash of each coordinate x, and y of the public-key point with the HMAC-SHA-1 hash function.

5.2 Parameters

We detail the value of the parameters used for our implementation in table 2.

Table 2: Accumulator parameter values

Accumulator	Parameter	Value
ECC	Curve domain param.	secp160r1, secp192r1
Bloom	Capacity False positive rate	100 2^{-90}

For Bloom case study we chose a network composed of 100 nodes. It can be considered as a small network of sensors. 2^{-90} is an acceptable false positive rate. We wanted to benchmark our implementation on large-scale network, but due to storage constraint we chose to use only 100 keys. [1]'s evaluation involved 1000 nodes. A 2^{-90}-false positive rate 1000-element Bloom filter approximately occupies up to 16KB that is impossible to store in RAM using our sensors.

[1] `https://github.com/vjutvik/Contiki-IPsec.git`

[2] `https://github.com/erimatnor/opsim/blob/ae42ee916d64a392410c4cdfc8c5213d2edbada4/bloomfilter.c`

6 Performance evaluation

In order to benchmark the performance of both accumulators we measured the time of authentication of a node, the size of the accumulators and the memory usage of programs implementing [1]'s protocol.

Table 3: Memory usage for the variables employed during the node-to-node key-verification (bytes)

Instance	WSN430			MB851	
	ECC		Bloom	ECC	
	secp160r1	secp192r1		secp160r1	secp192r1
PK	44	52	20	48	56
SK	22	26	20	24	28
z	44	52	1985	48	56
w	44	52	/	48	56

Table 3 shows the memory usage of the variable employed during the node-to-node key-verification. We observe that the Bloom filter occupies much more place in memory than the ECC-based accumulator. Moreover with nearly 2KB the Bloom filter takes up 20% of the RAM available on a WSN430 node. Besides, we failed to experiment a real exchange and authentication of keys between nodes using Bloom filter (however we succeeded to launch verifications on a single node without any network function). It must be reminded that ECC-based accumulator size is constant regardless the number of accumulated elements. Consequently, we can reasonably say that in terms of memory the ECC-based accumulator is better.

Table 4: Average key verification time in node-to-node key verification (seconds).

Sensor	ECC		Bloom
	secp160r1	secp192r1	
WSN430	3.88	5.49	0.33
MB851	0.45	0.55	/

Table 4 outlines the average time of the key verification. For ECC-based accumulators, 15 keys have been accumulated. With 100 keys, the Bloom filter appears to be ten times as faster than the ECC-based accumulator. Due to RAM constraints, we tested the ECC-based accumulator on MB851 with only one node and we disabled network functions on it. This is not a real problem since it exists sensors with the same processor and more embedded RAM. Verification times on MB851 are roughly similar to the times announced by [1] on Sun SPOT sensors. Times observed are very promising. It demonstrates that it is totally possible to use ECC-based accumulators on low-power devices. Node occupancy time needed for the computations is acceptable. As a consequence energy consumption will be satisfactory too.

Table 5: ECC-based accumulator memory usage in node-to-node key-verification programs (bytes)

	WSN430			MB851		
	hello-world	secp160r1	secp192r1	hello-world	secp160r1	secp192r1
RAM	4600	6258	6558	5064	6220	6396
Flash	19014	32846	33048	26460	34706	34900
Total	23614	39104	39606	31524	40926	41296

Table 5 reports the memory usage of our programs estimated with msp430−size and arm−none−eabi−size. msp430−size has already been used to profile memory usage of Contiki programs in [17]. The hello-world program corresponds to the example given in the Contiki repository. RAM equals to the sum of the size of the symbols in BSS and $Data$ segments and flash corresponds to the sum of symbol size in $Text$ and $Data$ areas. These estimates reflect the size in memory taken up by the ECC library, the ECC parameters (curve and keys), the accumulator, and the network functions. It definitely confirms that it is also possible to use [1]'s protocol on memory constrained devices.

7 Related work

During the last decade several papers about ECC in wireless sensor networks were published. [18] presents experiments on the elliptic curve point multiplication on 8-bit CPUs. This operation is another name for the elliptic curve scalar-point product, the one-way quasi-commutative function used in ECC-based accumulators. Their implementations are in assembly code. Time results vary from 0.81s to 4.58s for secp160r1 curve parameter set, and from 1.24s to 7.56s for secp192r1 curve parameter set. Observed times are in the same range as ours.

8 Conclusion and future work

Through our implementation we demonstrated that the distributed key certification protocol using accumulators works well on very constrained devices. ECC-based accumulator usage is practical both in terms of memory usage and execution time even on this kind of sensors. Bloom filter is not usable in large-scale sensor networks because of the size taken up by the accumulator, at least when the filter is stored in RAM. We could try to store it in flash memory. Nevertheless it should be noted that in a small network, computations with Bloom filters and convenient false positive rates are faster than with ECC-based accumulators. The next objective is to implement a complete protocol for secure communication establishment in wireless sensor networks.

Acknowledgments

The special thank goes to my supervisors Jun-Young Bae and Franck Rousseau. The supervision and support they provided me truly help the progression of this project and made me this work an enjoyable experience. My grateful thanks also go to Marie-Paule Cani who organizes this module of research for the Ensimag students.

References

[1] Jun-Young Bae, Franck Rousseau, Claude Castelluccia, and Cédric Lauradoux. Distributed key certification using accumulators for wireless sensor networks. Article not pusblished, Poster : `http://semba.conf.citi-lab.fr/programme/Documents/Posters/Poster-Jun_Young_BAE.pdf`. [Online, accessed: 2013].

[2] John Zachary. A decentralized approach to secure management of nodes in distributed sensor networks. In *Military Communications Conference, 2003. MILCOM'03. 2003 IEEE*, volume 1, pages 579–584. IEEE, 2003.

[3] Moteiv Corporation. Tmote sky : Datasheet. `http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf`.

[4] Oracle Labs. Sun spot sdk release notes. http://www.sunspotworld.com/docs/Yellow/ReleaseNotes.html. [Online, accessed: 08/02/2013].

[5] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[6] Victor S Miller. Use of elliptic curves in cryptography. In *Advances in CryptologyCRYPTO85 Proceedings*, pages 417–426. Springer, 1986.

[7] Recommended elliptic curve domain parameters, September 2000.

[8] Josh Benaloh and Michael De Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Advances in CryptologyEUROCRYPT93*, pages 274–285. Springer, 1994.

[9] Dae Hyun Yum, Jae Woo Seo, and Pil Joong Lee. Generalized combinatoric accumulator. *IEICE transactions on information and systems*, 91(5):1489–1491, 2008.

[10] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[11] Elaine Barker, Don Johnson, and Miles Smid. Nist special publication 800-56a recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revised). Technical report, Tech. rep., National Institute of Standards and Technology, 2007.

[12] Contiki os website. http://www.contiki-os.org.

[13] Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali. Protothreads: simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 29–42. Acm, 2006.

[14] Senslab platform. http://www.senslab.info.

[15] Kasun Hewage. Contikiecc. http://score.ucsc.lk/projects/contikiecc/wiki.

[16] Shahid Raza, Thiemo Voigt, and Vilhelm Jutvik. Lightweight ikev2: A key management solution for both the compressed IPsec and the IEEE 802.15. 4 security. In *Proceedings of the IETF Workshop on Smart Object Security*, 2012.

[17] Lander Casado and Philippas Tsigas. Contikisec: A secure network layer for wireless sensor networks under the contiki operating system. In *Proceedings of the 14th Nordic Conference on Secure IT Systems: Identity and Privacy in the Internet Age*, NordSec '09, pages 133–147, Berlin, Heidelberg, 2009. Springer-Verlag.

[18] 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 2004*, pages 119–132. Springer, 2004.