NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks

Piotr Szczechowiak[1], Leonardo B. Oliveira[2,⋆], Michael Scott[1], Martin Collier[1], and Ricardo Dahab[2]

[1] Dublin City University, Ireland
piotr@eeng.dcu.ie, mike@computing.dcu.ie, collierm@eeng.dcu.ie
[2] UNICAMP, Brazil
leob@ic.unicamp.br, rdahab@ic.unicamp.br

Abstract. By using Elliptic Curve Cryptography (ECC), it has been recently shown that Public-Key Cryptography (PKC) is indeed feasible on resource-constrained nodes. This feasibility, however, does not necessarily mean attractiveness, as the obtained results are still not satisfactory enough. In this paper, we present results on implementing ECC, as well as the related emerging field of Pairing-Based Cryptography (PBC), on two of the most popular sensor nodes. By doing that, we show that PKC is not only viable, but in fact attractive for WSNs. As far as we know pairing computations presented in this paper are the most efficient results on the MICA2 (8-bit/7.3828-MHz ATmega128L) and Tmote Sky (16-bit/8.192-MHz MSP-430) nodes.

Keywords: Wireless Sensor Networks, Elliptic Curve Cryptography, pairings, cryptographic primitives, implementation.

1 Introduction

Wireless sensor networks (WSNs) are ad hoc networks comprised mainly of small sensor nodes with limited resources and one or more base stations (BSs). Usually a BS is a much more powerful laptop-class node that connects the sensor nodes to the rest of the world [1,2]. WSN's are used for monitoring purposes, and provide information about the area being monitored to the rest of the system. Application areas range from battlefield reconnaissance and emergency rescue operations to surveillance and environmental protection.

Like any wireless ad hoc network, WSNs are vulnerable to many different attacks [3,4]. Besides the well-known vulnerabilities due to wireless communication and their distributed nature, WSNs face additional problems. Sensor nodes are usually small, cheap devices that are unlikely to be made tamper-resistant or tamper-proof and after deployment they are left unattended which makes them easily accessible to malicious parties. It is therefore crucial to add security

⋆ Supported by CAPES (Brazilian Ministry of Education) grant 4630/06-8 and FAPESP grant 2005/00557-9.

R. Verdone (Ed.): EWSN 2008, LNCS 4913, pp. 305–320, 2008.

to WSNs, especially in those applications where nodes are distributed in open environments.

Until recently it used to be thought that Public-Key Cryptography (PKC) was impractical in resource-constrained nodes and that security primitives must depend only on symmetric cryptosystems (e.g., RC5 [5] and SkipJack [6]). Although more efficient than PKC, symmetric cryptosystems suffer from some drawbacks (e.g., the *key distribution problem*[1]) which make them not well-suited for every WSN application.

This fact has motivated work on how to compute PKC efficiently in sensor nodes (e.g., [7,8,9,10]). The problem is challenging because those tiny devices have very limited battery life and we cannot afford to spend too much processor time on additional computations. By using Elliptic Curve Cryptography (ECC) [11,12] it has been shown (e.g., [8,9]) that PKC is indeed feasible in WSNs. This is because ECC demands considerably less resources than more conventional PKC (e.g. RSA/DSA), for a given security level. This feasibility, however, does not necessarily mean attractiveness, as the results presented so far are still too time consuming for some applications.

In this paper, we present updated results on implementing ECC, and PBC, over two of the most popular WSN platforms. By doing that, we show that these types of PKC are not only viable, but in fact attractive for resource-constrained sensor nodes. More specifically, we present results on computing point multiplication and pairings over MICA2 and Tmote Sky nodes. Our main contributions are

1. To show that ECC and PBC based PKC is not only viable, but in fact efficient for resource-constrained nodes;
2. To present the first known implementation of pairings over binary field for sensor networks.

Our code is based on Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL) [13], which is a publicly available library written in C, and thus can be easily ported to other devices.

The remainder of this work is organized as follows. In Section 2, we discuss related work. In Section 3, we introduce some basic ECC and PBC concepts. Implementation issues and performance results are presented in Sections 4 and 5, respectively. Finally, we conclude in Section 6.

2 Related Work

WSNs are a subclass of MANETs, and much work (e.g., [14,15]) has been proposed for securing MANETs in general. These studies are not applicable to WSNs because they assume laptop or palmtop-level resources, which are orders of magnitude larger than those available in WSNs. Conventional public key

[1] The key distribution problem is the problem of how to set up secret keys between communicating nodes.

based methods are an example of a type of application which, while practical in a MANET, would be impracticable for a WSN.

Among the studies specifically targeted to resource-constrained WSNs, some (e.g,[4,3]) have focused on attacks and vulnerabilities. Wood and Stankovic [4] surveyed a number of denial of service attacks against WSNs, and discussed some possible countermeasures. Karlof and Wagner [3] focused on routing layer attacks, and showed how some of the existing WSN protocols are vulnerable to these attacks.

Many security proposals for WSNs (e.g., [5,16,17,18,19,20,21,22,23,24]) have focused on efficient key management of symmetric encryption schemes. Perrig *et al.* [5] proposed SPINS, a suite of efficient symmetric key based security building blocks. Eschenauer *et al.* [16] looked at random key predistribution schemes, which provoked a large number of follow-on studies [25]. In [17] Zhu *et al.* proposed LEAP, a rather efficient scheme based on local distribution of secret keys among neighboring nodes.

The studies specifically targeted to PKC have tried either to adjust conventional algorithms (e.g. RSA) to sensor nodes, or to employ more efficient techniques (e.g. ECC) in this resource-constrained environment. All the seminal papers of Watro *et al.* [7], Gura *et al.* [8], and Malan *et al.* [9] have used the ATmega128L microprocessor as the implementation platform. Watro *et al.* [7] proposed TinyPK. To perform key distribution, TinyPK assigns RSA efficient public operations to nodes and RSA expensive private operations to better equipped external parties. Gura *et al.* [8] reported results for ECC and RSA primitives on the ATmega128L and demonstrated convincingly that the former outperforms the latter. Their ECC implementation is based upon arithmetic in the prime finite field $GF(p)$. In order to speed up integer multiplication in this field they came up with the idea for the hybrid multiplication. In our work we exploit improvements to this method to make it even more efficient on both considered platforms. Malan *et al.* [9] have presented the first ECC implementation over binary fields $GF(2^m)$ for sensor nodes. They used polynomials basis and presented results for the ECDH key exchange protocol. More recently, Liu *et al.* developed TinyECC [26], an ECC library that provides elliptic curve arithmetic over prime fields and uses inline assembly code to speed up critical operations on the ATmega128 processor. Lately, they have also added support for the MSP430 and XScale platforms.

Some of the research in cryptographic implementation has focused specifically on the MSP430 processor. Guajardo *et al.* [27] have shown that scalar point multiplication over prime fields can be achieved efficiently without any stored/precomputed values. They used the MSP430x33x family of microprocessors which is not used in current WSN motes. Wang *et al.* [28] worked with the TelosB mote [29], which also features the MSP430 processor. They presented results for basic ECC operations over prime fields, such as point addition, point doubling and point multiplication.

In the literature we can find some papers (e.g [30,31,32,33,34,35,10]) that envision WSNs as a scenario in which to exploit Pairing-based Cryptography (PBC).

Of those related to pairing implementation on sensor nodes, McCusker *et al.* [33] have focused on a hardware solution that both implements primitives for computing the Tate pairing and meets the strict energy constraints of sensor nodes. Doyle *et al.* [32] presented simulation results on pairings on the ARM7TDMI [36] processor. This platform, however, is considerably more powerful than any of the devices that are used in WSN's at the moment. In [35] Oliveira *et al.* focused on the ATmega128L and described the possibility of implementing the Tate pairing on this platform. Nevertheless no actual implementation was presented. Finally, Oliveira *et al.* [10] recently presented TinyTate and showed that software implementation of PBC is indeed viable in resource-constrained nodes, even though its level of security was not adequate for all applications. TinyTate also targets the ATmega128L and uses TinyECC as the underlying library.

3 Concepts

ECC was independently introduced by Miller [11] and Koblitz [12]. As opposed to conventional PKC (e.g. RSA/DSA), there is no sub-exponential algorithm known to solve ECC's underlying hard problems and ECC can thus offer equivalent security using smaller parameters [37].

Cryptography using Pairings (PBC), on the other hand, is an emerging field related to ECC which has been attracting the interest of international cryptography community, since it enables the design of original cryptographic schemes and makes well-known cryptographic protocols more efficient. Pairings, such as the Weil pairing, were first used in the context of cryptanalysis [38], but their first use in cryptography is due to the works of Sakai [39] *et al.* and Joux [40].

In this section we briefly introduce some ECC and PBC concepts. For more information on this issues please refer to, for instance, López *et al.* [37] and Galbraith [41]. In what follows, let $E/\mathbb{F}_q$ be an elliptic curve over a finite field $\mathbb{F}_q$, and $E(\mathbb{F}_q)$ be the group of points on this curve, and $\#E(\mathbb{F}_q)$ be the group order.

Bilinear Pairing. Let ℓ be a positive integer. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be additively-written groups of order ℓ with identity $\mathcal{O}$, and let $\mathbb{G}_T$ be a multiplicatively-written group of order ℓ with identity 1.

A *bilinear pairing* is a computable, non-degenerate function

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T.$$

The most important property of pairings in cryptographic constructions is the bilinearity, namely:

$$\forall\, P \in \mathbb{G}_1, \forall\, Q \in \mathbb{G}_2 \text{ and } \forall\, a, b \in \mathbb{Z}^*, \text{ we have}$$

$$e([a]P, [b]Q) = e(P, [b]Q)^a = e([a]P, Q)^b = e(P, Q)^{ab}.$$

In practice, the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are implemented using a group of points on certain special elliptic curves and the group $\mathbb{G}_T$ is implemented using a multiplicative subgroup of an extension of the underlying finite field. For certain families of supersingular elliptic curves we have $\mathbb{G}_1 = \mathbb{G}_2$.

Discrete Logarithm Problem. Let $\mathbb{G} = \langle \alpha \rangle$ be multiplicatively-written group of order n with generator α and let β be an element of $\mathbb{G}$. The Discrete Logarithm Problem (DLP) is to compute an integer l such that $\beta = \alpha^l$.

Elliptic Curve Discrete Logarithm Problem. Elliptic Curve Discrete Logarithm Problem (ECDLP) is: given a point P of order n and $Q \in \langle P \rangle$, compute $l \in [0, n-1]$ such that $Q = lP$.

Elliptic Curve Diffie-Hellman Problem. Elliptic Curve Diffie-Hellman Problem (ECDHP) is: given a point P, $[a]P$, and $[b]P$ for some $a, b \in \mathbb{Z}^*$, compute $[ab]P$.

Embedding Degree. A subgroup $\mathbb{G}$ of $E(\mathbb{F}_q)$ is said to have an *embedding degree* k with respect to ℓ if k is the smallest integer such that $\ell \mid q^k - 1$.

Bilinear Diffie-Hellman Problem. Most of the pairing applications rely on the hardness of the following problem for their security [41]: given P, $[a]P$, $[b]P$, and $[c]P$ for some $a, b \in \mathbb{Z}^*$, compute

$$e(P, P)^{abc}.$$

This problem is known as the *Bilinear Diffie-Hellman Problem*. The hardness of the Bilinear Diffie-Hellman Problem depends on the hardness of the Diffie-Hellman problems both on $E(\mathbb{F}_q)$ and in $\mathbb{F}_{q^k}$. So, for most pairing applications the parameters q, ℓ, and k must satisfy the following security requirements:

1. ℓ must be large enough so that solving the ECDLP in an order-ℓ subgroup of $E(\mathbb{F}_q)$ is infeasible (e.g. using Pollard's rho algorithm);
2. k must be large enough so that solving the DLP in $\mathbb{F}_{q^k}$ is infeasible (e.g., using the index-calculus method).

The Tate Pairing. Let $E(\mathbb{F}_q)$ contain a subgroup of prime order ℓ coprime with q and with embedding degree k. (In most applications, ℓ is also a large prime divisor of $\#E(\mathbb{F}_q)$.) The *Tate pairing* is the bilinear pairing

$$\hat{e} : E(\mathbb{F}_{q^k})[\ell] \times E(\mathbb{F}_{q^k})/[\ell]E(\mathbb{F}_{q^k}) \rightarrow \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^{\ell}.$$

4 Implementation

Our implementation of Elliptic Curve primitives targets two different platforms the 8-bit Atmel ATmega128L and the 16-bit Texas Instruments MSP430F1611, as these are most commonly used processors in Wireless Sensor Network nodes nowadays. Although both microcontrollers have RISC architectures, they differ in many ways. ATmega128L [42] has a very modern advanced RISC architecture where most of 133 instructions are executed in a single clock cycle. In contrast, the MSP430F1611 [43] has a more traditional architecture, offers 27 instructions in 7 addressing modes and uses mainly memory based operations. The former

CPU has 32 8-bit registers and three 16-bit pointer registers, the latter provides 16 16-bit registers from which only 12 are available for general purpose use. The Atmel product operates at 7.3828 MHz, and offers 4KB of RAM memory and 128 KB of program space, whereas MSP430 has 8.192 MHz frequency, 48 KB ROM and 10 KB RAM. The Texas Instruments microcontroller also embeds a 16×16 bit hardware multiplier and has an ultra-low power design, which makes it more efficient in terms of current consumption than the ATmega128L.

We chose two popular WSN motes to test the performance of our ECC based programs: The MICA2 [44] platform developed by Crossbow Technology and the Tmote Sky [45] developed by Moteiv corporation. MICA2 mote is build upon the ATmega128L processor, incorporates a 433 MHz radio and has 512 KB of FLASH memory to store measurement data. The Tmote Sky uses the MSP430F1611 microcontroller and Chipcon CC2420 wireless transceiver which operates in the 2.4 GHz ISM band and provides transmission speeds up to 250 Kbps. Using the TinyOS [46] operating system allowed us to run the same programs written in nesC [47] language on both WSN motes. Porting the code from one device to another was a lot easier with the use of TinyOS, which enables the use of features like timers, I/O interfaces, LED's, etc. in an unified way. This approach allowed us to hid most of the hardware dependencies for different platforms and simplified the programming.

Almost all of our code was written in the C/nesC language and can be easily ported to other 8, 16 or even 32-bit resource constrained platforms. This approach is a trade-off between size and re-usability of the code. In order to speed up the execution of particularly time-critical functions we have replaced standard C code with some assembly language specific for each platform. Even though we used inline assembly in our programs we made the whole process as portable as possible. Our assembler routines were generated automatically by special utility program from user defined macros. In this simple and convenient way appropriate assembler code can be quickly developed for new platforms and processors that are not yet supported.

4.1 Basic Primitives Implementation

NanoECC is based on MIRACL [13] (Multiprecision Integer and Rational Arithmetic C/C++ Library) which provides all the necessary Elliptic Curve primitives and functions to compute Pairings and to implement protocols like ECDH, ECDSA. MIRACL is a set of tools that supports standard symmetric-key and public-key cryptography. It handles big numbers arithmetic and offers full support for Elliptic Curve Cryptography over the prime field $GF(p)$, and the binary field $GF(2^m)$. MIRACL is also a good choice when implementing cryptographic services in an embedded environment. It has built in features that allows to run ECC even on very constrained environments and tiny architectures which do not support a heap. In this case all memory can be allocated exclusively from the stack. This allows maximum use and re-use of memory, and avoids fragmentation of precious RAM. We have optimized MIRACL library to achieve the best ECC performance on our WSN motes.

Different Elliptic Curve Cryptography implementations for sensor networks [27], [26], [28], [8] were mainly focused on the prime finite fields. The choice of the field was dictated by the fact that basic arithmetic operations can be effectively optimized if pseudo-Mersenne primes are used in $GF(p)$. $GF(2^m)$ fields were not favoured because binary polynomial arithmetic (multiplication in particular) is insufficiently supported by current CPU's. This paper compares results achieved using both types of finite fields and shows that in some cases in this constrained environments, ECC operations over binary polynomial field $GF(2^m)$ outperforms those in $GF(p)$. On top of that timings for binary field case would be significantly faster if a "binary polynomial multiplication" instruction was available on the considered architectures.

Modular arithmetic routines are fundamental operations in every Elliptic Curve system. The overall performance of ECC depends greatly on the speed of those primitives. In $GF(p)$ big integer multiplication and reduction modulo p of the result are the most time-critical operations and must be performed as quickly as possible. We have used a variant of the hybrid multiplication method proposed in [8] to achieve this goal. Our implementation minimizes the number of operations on memory and uses additional CPU registers for catching and storing the carry bits. The Hybrid method takes advantage of extra registers to avoid unnecessary load operations and becomes more efficient with the number of registers used. On ATmega128L we were able to implement hybrid multiplication with column size $d = 4$. Due to the small number of general purpose registers on MSP430F1611 we could only achieve $d = 2$, using all 12 available registers. For more details concerning the implementation of our improved hybrid method see the paper [48]. For modular reduction a fast algorithm was implemented that takes advantage of special form of $p = 2^{160} - 2^{112} + 2^{64} + 1$ (a Solinas prime) as the modulus, using a 160-bit Elliptic Curve. Multiplication and reduction along with squaring, modular addition and modular subtraction were all implemented in assembly language. All the results for those routines in instruction cycles, assuming 160-bit integer operands, are listed in Table 1. Taking into account the 7.3828 MHz clock on the ATmega128L and the 8.192 MHz clock on Tmote Sky, 160-bit numbers multiplication can be performed in 0.36 ms and 0.21 ms respectively.

The field $GF(2^m)$ is usually constructed using a polynomial basis representation. In this case binary polynomials multiplication and reduction modulo

Table 1. Timings in instruction cycles for basic modular arithmetic routines using 160-bit integers on ATmega128L and MSP430F1611

	ATmega	**MSP430**
hybrid multiplication	2654 (d=4)	1746 (d=2)
squaring	2193 (d=4)	1373 (d=2)
modular reduction	1228	990
modular addition	340-470	105-235
modular subtraction	340-470	105-235

an irreducible binary polynomial are the crucial operations. As described in [49] Karatsuba-Ofman multiplication algorithm were adopted for the polynomial case. This divide-and-conquer technique allowed us to reduce multiplication complexity by using word size polynomial multiplication and extra additions (which are very fast in $GF(2^m)$). Throughout our research we observed that assembler implementation of binary polynomials multiplication did not improve our timings as much as we had expected. This was due to the fact that binary polynomial word multiplication and variable length shifts instructions were not available on our target devices. In the end we decided to implement all binary field primitives and operations using the standard C language. In order to speed up reduction routines on both platforms we have developed fast field-specific code for the reduction modulo the irreducible polynomials $x^{163} + x^7 + x^6 + x^3 + 1$ (as recommended by NIST) and $x^{271} + x^{201} + 1$. For guidance on the optimal irreducible polynomials for $GF(2^m)$ arithmetic to use in given circumstances please refer to [50].

4.2 ECC Implementation

One difficulty in using ECC is that of finding a suitable Elliptic Curve. Curve parameters have to be carefully chosen to allow efficient computations and provide a reasonable level of security. NIST recommends using at least 160-bit keys in ECC systems to achieve security level equivalent to that offered by standard RSA based solutions with 1024-bit keys. In our example programs we decided to use NIST k163 Koblitz curve over $GF(2^{163})$ binary field and $y^2 = x^3 - 3x + 157$ curve with $p = 2^{160} - 2^{112} + 2^{64} + 1$ over $GF(p)$. The usage of Koblitz curve gives a significant speed up when performing a point multiplication, as no expensive point doublings are required. To satisfy security requirements mentioned in section 3 our pairing parameters were chosen as $k \cdot \ell > 1024$. In this inequality k stands for embedding degree and ℓ stands for number of bits in p (in the case of prime fields $GF(p)$) or m (in case of binary fields $GF(2^m)$). To satisfy these conditions we chose the supersingular curve $y^2 + y = x^3 + x^2$ with $k = 4$ and $x^{271} + x^{201} + 1$ as the reduction polynomial for the binary field. For the pairing program in the prime field we used a $y^2 = x^3 + Ax + B$ curve, $k = 4$ and 256-bit modulus p. Parameter A was set to -3 in order to reduce the number of operations for the point doubling routine.

ECC operations are based on arithmetic involving the points of the elliptic curve and as mentioned before it is essential to optimize basic arithmetic operations in underlying fields. Overall performance of the system is also highly dependably on efficient implementation of curve operations. Two of those fundamental operations are point addition and point doubling. Please see [49] for a geometrical explanation of those operations. The selection of points coordinate system has a big influence on the performance of the above mentioned operations. It has been shown that projective coordinate systems (x, y, z) are more efficient than affine (x, y) systems. Rules for point addition and point doubling in affine coordinates requires inversion in underlying field, which is usually much more expensive than multiplication. The same operations in projective coordinates uses

greater number of cheaper multiplications and squarings in place of an inversion and thus makes it more suitable to our target platforms.

A common operation in ECC is the computation of sP, where s is an integer and P is a point on an elliptic curve. This operation is called point multiplication and can be decomposed into a sequence of point additions and point doublings. These operations dominate the overall execution time of elliptic curve cryptographic schemes, and so optimization is important. The ECDH and ECDSA protocols require multiplication by a scalar of a fixed base point on the selected curve, and this can be carried out more quickly using precomputation. Our example programs therefore implement a fixed point multiplication method using additional storage to accelerate the calculations. The Comb method for point multiplication described in [49] was used in this case. Precomputation was performed with window size $w = 4$ resulting in 16 elliptic curve points stored in ROM. With this approach point multiplication is a tradeoff between memory space and computation time.

Pairing based systems have become more and more popular in Public Key Cryptography schemes. At first it appeared that these operations are far too complex to be calculated in reasonable amount of time on tiny architectures like WSN nodes. However our implementation shows that pairings can by computed quickly and efficiently on small and constrained devices such as MICA2 or Tmote Sky. The Tate pairing denoted as $e(P, Q)$, on an elliptic curve $E(\mathbb{F}_{q^k})$, evaluates as an element of an extension field $\mathbb{F}_{q^k}$. This requires implementation of extension field arithmetic routines. We used $k = 4$, so special procedures for multiplication, squaring, exponentiation, inversion and calculation of square roots in quadratic extension fields ($\mathbb{F}_{p^4}$ and $\mathbb{F}_{2^{4m}}$) were developed. More detailed descriptions of those routines can be found in [51].

There has been a lot of work on efficient implementation of pairings on elliptic curves. This research shows that some of the best results can be achieved on supersingular curves over fields of low characteristic. For this reason we chose the improved Duursma-Lee algorithm to compute the Tate pairing over $GF(2^m)$ based on η_T pairing, which is one of the fastest known. Due to space limitations please refer to [52] for a detailed description and explanation of this method. The pairing operation in the prime field on MSP430 was implemented and optimized as described in [53] and is based on Miller's algorithm. We used a different approach on the ATmega128L, where more program space was available. Here we have implemented the Ate pairing on a non-supersinglar curve over $GF(p)$ as described in [54] with parameters $k = 4$, p a 256-bit prime, and a fixed point P. Knowing P's coordinates allowed us to use precomputation, which speeds up elliptic curve point multiplication. In the next section we will evaluate NanoECC and show results for computing pairings and point multiplications.

5 Results

Efficient implementation of ECC on such constrained devices as WSN nodes is not an easy task. Issues like the small amount of memory, limited CPU

capabilities and scarce battery resources have to be taken into consideration. Program code needs to be highly optimized to meet all those demands. That is the reason why there is some confusion in the literature concerning timings of basic Elliptic Curve operations on those constrained platforms. The results presented in recent papers vary a lot. In our research we have tested the limits of ECC in sensor networks and our results give a clear answer to the question of how long Elliptic Curve Cryptography primitives take on standard WSN motes.

NanoECC is optimized for speed. Memory usage was our secondary concern as optimizing for code size lowers functionality and portability due to greater number of assembly routines. A large set of available library functions in NanoECC gives a lot of flexibility in writing ECC based programs. Most of the procedures were developed using standard C which favors speed and allows us to re-use the code on numerous other WSN platforms.

5.1 Point Multiplication

Our example programs were compiled under TinyOs operating system and run on the MICA2 and Tmote Sky motes, so all measurements were taken on actual devices. We decided to measure three most important parameters for sensor nodes: computation time, memory usage and energy consumption. Both devices had to be slightly modified to facilitate data acquisition. A precise one ohm resistor was soldered between the mote and its battery pack to measure the exact amount of current drained during program execution. Input/Output ports on MICA2 and Tmote Sky were used to pass trigger signals to the measuring device. We used National Instruments NI 5112 digitizer card to acquire measurement data from both nodes. In this way exact timings and precise power consumption information could be gathered without using the mote's timers and other features which increase computation overhead.

All experiments were carried out using LabVIEW software. Figure 1 shows an example graph of Tmote Sky voltage levels during Elliptic Curve Diffie Helman (ECDH) program execution. As we can see point multiplication takes a considerable percentage of the total duration of the program. Current drawn from the battery pack was calculated based on voltage levels. The average value of current consumption was taken from all the samples within the program execution period. Both motes were powered with two AA batteries, so a voltage equal approximately to 3 V was provided (assuming new batteries). Based on this information total energy consumption was calculated from the formula $E = U \cdot I \cdot T$, where T is code running time.

Results for point multiplication operation in binary and prime fields on MICA2 and Tmote Sky nodes are listed in Table 2. Precomputation was used in all cases to speed up the point multiplication routine. Point addition and point doubling operations were not considered independently because their computation time is insignificant comparing to point multiplication. As we can see point multiplication is faster on both platforms when using the prime field, but the difference is not as big as we might have expected. Results in $GF(2^m)$ are comparable to those in $GF(p)$, even though both CPU's don't support special

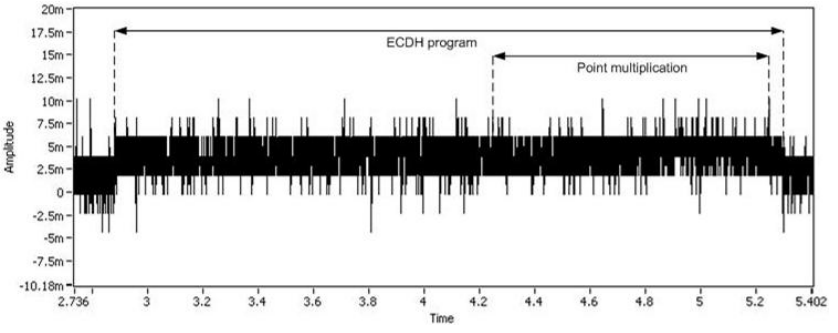

Fig. 1. Voltage levels on Tmote Sky during example ECDH program execution

Table 2. Performance evaluation of point multiplication on MICA2 and Tmote Sky

	MICA2		**Tmote Sky**	
	Binary field	Prime field	Binary field	Prime field
Computation time	2.16s	1.27s	1.04s	0.72s
Current draw	7.86mA	7.88mA	3.45mA	3.68mA
Energy consumption	50.93mJ	30.02mJ	10.76mJ	7.95mJ
ROM	32.4KB	46.1KB	32.1KB	31.3KB
RAM	1.7KB	1.8KB	2.8KB	2.9KB

instructions for binary field arithmetic. Achievements in binary field are even more competitive because no assembly language routines were used in this case. We used the hardware multiplier on Tmote Sky's CPU to improve timings for big numbers multiplication in the prime field. This fact has an influence on average current consumption, which is slightly higher when the multiplier unit is turned on. Operations in binary field of course do not require the hardware multiplier. On the MICA2 the average current drawn is almost the same using both fields. Total energy consumption in all experiments was lower for the prime field case, because of the faster execution time of point multiplication. Looking at the power consumption on both platforms, it is clear that Tmote Sky is far more efficient using even in some cases 5 times less energy for the same work carried out.

Program size figures given in Table 2 include only our ECC implementation without counting additional storage for TinyOs modules. The numbers for the RAM memory requirement were not taken directly from TinyOs output, because they did not include stack usage. Simulation environments such as AVR Studio and IAR Embedded Workbench for MSP430 allowed us to achieve precise information about RAM usage and stack size at any given time during our programs execution. Figures for RAM, presented in Table 2 show the maximum usage that we have encountered. Average RAM utilization was usually much lower than that.

Our results for point multiplication on Tmote Sky compares favourably with numbers presented in [28]. Their experiments were performed on TelosB mote which has exactly the same processor as Tmote Sky. In our experiments fixed point multiplication on 160-bit elliptic curve was performed in 0.72s which is a nice improvement comparing to 3.13s in [28]. Guajardo *et al.* [27] also tested point multiplication performance on tiny architectures and achieved 3.4s at 1MHz on MSP430x33x family of devices. This result cannot be directly compared with our achievements because they used a 128-bit prime in their implementation, which is not as secure as the 160-bit prime recommended by NIST. For point multiplication in the binary field on the MSP430F1611 processor our results are superior to those achieved in [55]. Their point multiplication in 163-bit finite field takes 32.5s which is quite a lot comparing to 1.04s in our implementation.

There were several attempts to implement ECC on MICA2 platform. Point multiplication in prime field was implemented by Gura *et al.* on ATmega128 [8]. They calculated point multiplication in 0.81s at 8MHz on secp160r1 curve. In our implementation same operation using a different curve takes 1.27s at 7.3828MHz on MICA2 mote. Malan *et al.* [9] implemented point multiplication in the binary field. However the result of 34.16s for this operation on a 163-bit curve is far from being optimal. In [56] the authors managed to perform fixed point multiplication in 6.74s on MICA2 but they used $GF(2^{113})$ arithmetic which should be much faster to calculate. Computation of the same routine in $GF(2^{163})$ in our implementation takes only 2.16s. To our knowledge, point multiplication results in binary field reported in this paper are the fastest known so far.

5.2 Pairing Evaluation

Table 3 shows all the results for pairing computation achieved on the MICA2 and Tmote Sky motes. Our timings show that pairing calculation can be performed in as fast as 5.25s on a resource constrained WSN node. As we can see pairing programs in binary field are much faster than in prime field on both our research platforms. The difference is quite significant, as much as 7s. Binary field pairings are also more efficient in terms of energy consumption and program size. Bigger code size for pairings in the prime field is due primarily to precomputation data. It is especially visible for MICA2 where constant precomputed values take 28K of a total 71.9K of program memory. We couldn't use that much of precomputation on Tmote Sky due to the 48KB memory limit. Otherwise the pairing program

Table 3. Results for pairing implementation on MICA2 and Tmote Sky

	MICA2		**Tmote Sky**	
	Binary field	Prime field	Binary field	Prime field
Computation time	10.96s	17.93s	5.25s	11.82s
Current draw	7.86mA	7.88mA	3.45mA	3.68mA
Energy consumption	258.44mA	423.87mJ	54.34mJ	130.49mA
ROM	53.5KB	71.9KB	30.3KB	47.0KB
RAM	2.8KB	2.5KB	3.7KB	3.0KB

on this platform would have been a bit faster. On the other hand RAM usage was a critical issue on ATmega128L. All variables and runtime objects had to be handled very carefully to fit 4KB of dynamic memory. All of our binary field pairing programs have a big advantage that they do not need any precomputation at all.

Although much research has been carried out in PBC, very little attention has focused on implementing those operations on resource constrained devices. Apparently pairings were considered as too heavyweight for WSN nodes. The first pairing implementation in WSN's was performed by Oliveira *et al.* [10]. In that work a $k = 2$ Tate pairing with a 256-bit prime was implemented over a supersingular curve $y^2 = x^3 + x$. The timing for this operation was estimated as 30.21s on 7.3828MHz MICAz mote (also ATmega128L CPU). Our implementation of the Tate pairing in the prime field on the MICA2 outperforms that result with 17.93s and offers a much higher level of security using bigger parameters. All of our results for pairing implementation show that those operations can be performed in a reasonable amount of time on small and constrained devices. As far as we know our pairing programs are at the moment the fastest implementations on popular WSN motes. Nevertheless we are pretty sure that further optimizations are possible in terms of memory usage as well as execution time.

6 Conclusion

Recent results in WSN research area show that PKC based on elliptic curves is indeed feasible in those constrained environments. However the performance of many ECC implementations is still a disappointment in terms of running time and resources usage. This fact prevents ECC based security protocols from being used in certain applications. Our achievements presented in this paper prove that ECC operations can be performed in a quick and efficient way on popular sensor network platforms. As our contribution, we present updated results on computing elliptic curve point multiplication and pairings. We also show that ECC over prime field is not always the best option as pairings over $GF(2^m)$ seem to be more efficient on this type of architecture. PBC offers a flexible cryptographic primitive that can be used in many new security protocols. Fast pairing computation enables Identity Based Encryption and thus opens new ways for achieving security in sensor networks. Future work will address this issue and will deal with some problems that need to be solved in order to develop a complete security protocol.

References

1. Estrin, D., Govindan, R., Heidemann, J.S., Kumar, S.: Next century challenges: Scalable coordination in sensor networks. In: MobiCom 1999. Mobile Computing and Networking, Seattle, WA USA, pp. 263–270 (1999)
2. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless Sensor Networks: a survey. Computer Networks 38(4), 393–422 (2002)

3. Karlof, C., Wagner, D.: Secure routing in Wireless Sensor Networks: Attacks and countermeasures. Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols 1(2–3), 293–315 (2003) (Also apeared in 1st IEEE International Workshop on Sensor Network Protocols and Applications)
4. Wood, A.D., Stankovic, J.A.: Denial of service in sensor networks. IEEE Computer 35(10), 54–62 (2002)
5. Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, J.D.: SPINS: Security protocols for sensor networks. Wireless Networks 8(5), 521–534 (2002) (Also appeared in MobiCom 2001)
6. Karlof, C., Sastry, N., Wagner, D.: Tinysec: A link layer security architecture for Wireless Sensor Networks. In: 2nd ACM SensSys., pp. 162–175 (2004)
7. Watro, R.J., Kong, D., fen Cuti, S., Gardiner, C., Lynn, C., Kruus, P.: Tinypk: securing sensor networks with public key technology. In: SASN 2004. 2nd ACM Workshop on Security of ad hoc and Sensor Networks, Washington, DC, pp. 59–64 (2004)
8. Gura, N., Patel, A., Wander, A., Eberle, H., Shantz, S.C.: Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 119–132. Springer, Heidelberg (2004)
9. Malan, D.J., Welsh, M., Smith, M.D.: A Public-Key Infrastructure for key distribution in TinyOS based on Elliptic Curve Cryptography. In: SECON 2004. 1st IEEE Intl' Conf. on Sensor and Ad Hoc Communications and Networks (2004)
10. Oliveira, L.B., Aranha, D., Morais, E., Daguano, F., López, J., Dahab, R.: TinyTate: Computing the TinyTate in resource-constrained nodes. In: 6th IEEE International Symposium on Network Computing and Applications, Cambridge,MA (2007)
11. Miller, V.: Uses of elliptic curves in cryptography, advances in cryptology. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
12. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of computation 48, 203–209 (1987)
13. Scott, M.: MIRACL—A Multiprecision Integer and Rational Arithmetic C/C++ Library. Shamus Software Ltd, Dublin, Ireland (2003), `http://www.shamus.ie`
14. Zhou, L., Haas, Z.J.: Securing Ad Hoc Networks. IEEE Network 13(6), 24–30 (1999)
15. Hubaux, J.P., Buttyán, L., Capkun, S.: The quest for security in mobile ad hoc networks. In: 2nd ACM international symposium on Mobile ad hoc networking & computing, pp. 146–155. ACM Press, New York (2001)
16. Eschenauer, L., Gligor, V.D.: A key management scheme for distributed sensor networks. In: CCS 2002. 9th ACM conf. on Computer and communications security, pp. 41–47 (2002)
17. Zhu, S., Setia, S., Jajodia, S.: LEAP: efficient security mechanisms for large-scale distributed sensor networks. In: CCS 2003. 10th ACM conference on Computer and communication security, pp. 62–72. ACM Press, New York (2003)
18. Pietro, R.D., Mancini, L.V., Mei, A.: Random key-assignment for secure Wireless Sensor Networks. In: SASN 2003. 1st ACM workshop on Security of ad hoc and sensor networks, pp. 62–71 (2003)
19. Kannan, R., Ray, L., Durresi, A.: Security-performance tradeoffs of inheritance based key predistribution for Wireless Sensor Networks. In: Castelluccia, C., Hartenstein, H., Paar, C., Westhoff, D. (eds.) ESAS 2004. LNCS, vol. 3313, Springer, Heidelberg (2005)

20. Çamtepe, S.A., Yener, B.: Combinatorial design of key distribution mechanisms for Wireless Sensor Networks. In: Samarati, P., Ryan, P.Y A, Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 293–308. Springer, Heidelberg (2004)
21. Liu, D., Ning, P., Li, R.: Establishing pairwise keys in distributed sensor networks. ACM Transactions on Information and System Security (TISSEC) 8(1), 41–77 (2005)(Also appeared in ACM CCS 2003)
22. Du, W., Deng, J., Han, Y.S., Varshney, P.K., Katz, J., Khalili, A.: A pairwise key pre-distribution scheme for Wireless Sensor Networks. ACM Transactions on Information and System Security 8(2), 228–258 (2005) (Also appeared in ACM CCS 2003)
23. Oliveira, L.B., Wong, H.C., Dahab, R., Loureiro, A.A.F.: On the design of secure protocols for hierarchical sensor networks. International Journal of Networks and Security (IJSN) 2(3/4), 216–227 (2007) (Special Issue on Cryptography in Networks)
24. Oliveira, L.B., Ferreira, A., cca, M.A.V., Wong, H.C., Bern, M., Dahab, R., Loureiro, A.A.F.: Secleach-on the security of clustered sensor networks. Signal Process 87(12), 2882–2895 (2007)
25. Hwang, J., Kim, Y.: Revisiting random key pre-distribution schemes for Wireless Sensor networks. In: 2nd ACM workshop on Security of ad hoc and sensor networks, pp. 43–52. ACM Press, New York (2004)
26. Liu, A., Kampanakis, P., Ning, P.: Tinyecc: Elliptic Curve Cryptography for sensor networks (ver. 0.3) (2007), `http://discovery.csc.ncsu.edu/software/TinyECC/`
27. Guajardo, J., Bluemel, R., Krieger, U., Paar, C.: Efficient implementation of Elliptic Curve Cryptosystems on the TI MSP430x33x family of microcontrollers. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, Springer, Heidelberg (2001)
28. Wang, H., Sheng, B., Li, Q.: Elliptic Curve Cryptography based access control in sensor networks. International Journal of Security and Networks (IJSN). Special Issue on Security Issues on Sensor Networks 1(3/4), 127–137 (2006)
29. Polastre, J., Szewczyk, R., Culler, D.: Telos: enabling ultra-low power wireless research. In: IPSN 2005. 4th international symposium on Information processing in sensor networks, p. 48. IEEE Press, Piscataway, NJ, USA (2005)
30. Zhang, Y., Liu, W., Lou, W., Fang, Y.: Securing sensor networks with location-based keys. In: WCNC 2005. IEEE Wireless Communications and Networking Conference (2005)
31. Oliveira, L.B., Dahab, R.: Pairing-based cryptography for sensor networks. In: 5th IEEE International Symposium on Network Computing and Applications, Cambridge, MA (fast abstract) (2006)
32. Doyle, B., Bell, S., Smeaton, A.F., McCusker, K., O'Connor, N.: Security considerations and key negotiation techniques for power constrained sensor networks. The Computer Journal 49(4), 443–453 (2006)
33. McCusker, K., O'Connor, N., Diamond, D.: Low-energy finite field arithmetic primitives for implementing security in Wireless Sensor Networks. In: 2006 Intl. Conf. on Communications, Circuits and systems. Computer, Optical and Broadband; Communications; Computational Intelligence, vol. III, pp. 1537–1541 (2006)
34. Bellare, M., Namprempre, C., Neven, G.: Unrestricted aggregate signatures. Cryptology ePrint Archive, Report 2006/285 (2006), `http://eprint.iacr.org/`
35. Oliveira, L.B., Dahab, R., Lopez, J., Daguano, F., Loureiro, A.A.F.: Identity-based encryption for sensor networks. In: PERCOMW 2007. 5th IEEE International Conference on Pervasive Computing and Communications Workshops, pp. 290–294 (2007)

36. Segars, S.: ARM7TDMI power consumption. IEEE Micro 17(4), 12–19 (1997)
37. López, J., Dahab, R.: An overview of Elliptic Curve Cryptography. Technical Report IC-00-10, Institute of Computing – UNICAMP (2000)
38. Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions on Information Theory 39(5), 1639–1646 (1993)
39. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: SCIS 2000. Symposium on Cryptography and Information Security, pp. 26–28 (2000)
40. Joux, A.: A one round protocol for tripartite diffie-hellman. J. Cryptology 17(4), 263–276 (2004) (Proceedings of ANTS-IV, 2000)
41. Galbraith, S.: Pairings. In: Blake, I., Seroussi, G., Smart, N. (eds.) Advances in Elliptic Curve Cryptography. London Mathematical Society Lecture Notes, pp. 183–213. Cambridge University Press, Cambridge (2005)
42. Atmel: ATmega128(L) datasheet (2006), `http://www.atmel.com`
43. TI: MSP 430F1611, Datasheet (2002), `http://www.ti.com`
44. Crossbow Technology, Inc. 41 Daggett Dr., San Jose, CA 95134: MPR/MIB Mote Hardware Users Manual – Document 7430-0021-05 (2003)
45. Moteiv: Tmote Sky datasheet (2006), `http://www.moteiv.com`
46. Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., Culler, D.: TinyOS: An operating system for Wireless Sensor Networks. In: Weber, W., Rabaey, J., Aarts, E. (eds.) Ambient Intelligence, Springer, New York (2004)
47. Gay, D., Levis, P., von Behren, J.R., Welsh, M., Brewer, E.A., Culler, D.E.: The nesC language: A holistic approach to networked embedded systems. In: ACM Conf. on Programming Language Design and Implementation, pp. 1–11 (2003)
48. Scott, M., Szczechowiak, P.: Optimizing multiprecision multiplication for Public Key Cryptography. Cryptology ePrint Archive, Report 2007/299 (2007)
49. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer, Heidelberg (2004)
50. Scott, M.: Optimal irreducible polynomials for $GF(2^m)$ arithmetic. Cryptology ePrint Archive, Report 2007/192 (2007)
51. Scott, M.: Implementing cryptographic pairings (2006)
52. Barreto, P.S.L.M., Galbraith, S., hEigeartaigh, C.O., Scott, M.: Efficient pairing computation on supersingular abelian varieties. In: Designs Codes And Cryptography, Boston/Norwell (USA) (2006)
53. Scott, M.: Computing the Tate Pairing. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 293–304. Springer, Heidelberg (2005)
54. Hess, F., Smart, N., Vercauteren, F.: The Eta Pairing revisited. IEEE Transactions on Information Theory 52(10), 4595–4602 (2006)
55. Arazi, O., Qi, H.: Load-balanced key establishment methodologies in Wireless Sensor Networks. International Journal of Security and Networks (IJSN). Special Issue on Security Issues on Sensor Networks 1(3/4), 158–166 (2006)
56. Blaß, E.O., Zitterbart, M.: Towards Acceptable Public-Key Encryption in Sensor Networks. In: The 2nd Int'l Workshop on Ubiquitous Computing, ACM SIGMIS (2005)