Elliptic Curve Based Zero Knowledge Proofs and Their Applicability on Resource Constrained Devices [*]

Ioannis Chatzigiannakis, Apostolos Pyrgelis, Paul G. Spirakis, Yannis C. Stamatiou
Research Academic Computer Technology Institute
and Computer Engineering and Informatics Department, University of Patras
Greece
{ichatz,pyrgelis,spirakis,stamatiou}@cti.gr

ABSTRACT

Elliptic Curve Cryptography (ECC) is an attractive alternative to conventional public key cryptography, such as RSA. ECC is an ideal candidate for implementation on constrained devices where the major computational resources i.e. speed, memory are limited and low-power wireless communication protocols are employed. That is because it attains the same security levels with traditional cryptosystems using smaller parameter sizes. Moreover, in several application areas such as person identification and eVoting, it is frequently required of entities to prove knowledge of some fact without revealing this knowledge. Such proofs of knowledge are called Zero Knowledge Interactive Proofs (ZKIP) and involve interactions between two communicating parties, the Prover and the Verifier. In a ZKIP, the Prover demonstrates the possesion of some information (e.g. authentication information) to the Verifier without disclosing it. In this paper, we focus on the application of ZKIP protocols on resource constrained devices. We study well-established ZKIP protocols based on the discrete logarithm problem and we transform them under the ECC setting. Then, we implement the proposed protocols on Wiselib, a generic and open source algorithmic library. Finally, we present a thorough evaluation of the protocols on two popular hardware platforms equipped with low end microcontrollers (Jennic JN5139, TI MSP430) and 802.15.4 RF transceivers, in terms of code size, execution time, message size and energy requirements. To the best of our knowledge, this is the first attempt of implementing and evaluating ZKIP protocols with emphasis on low-end devices. This work's results can be used from developers who wish to achieve certain levels of security and privacy in their applications.

Keywords

zero-knowledge proofs; elliptic curve cryptography; resource constrained devices; wireless communication;

1. INTRODUCTION

Recent advances in wireless communications and microelectro systems have lead to the construction of tiny devices with strong processing and communication capabilities. Mobile phones, PDAs, sensor devices and RFID tags are becoming a part of our daily lives and their networked interconnection makes the vision of the Internet of Things a real situation. Smart towns and buildings, automated hospitals, intelligent cars and sensors embedded in clothes are concepts that are already being developed.

However, the wireless nature of communication that these devices provide (e.g. 802.11, 802.15.4, Bluetooth) and the fact that there is not a fixed infrastructure in such dynamic networks raise significant security and trust issues [13]. In some cases, these petit computers may need to exchange crucial information that needs to be protected. Adversaries equipped with strong computers and antennae can eavesdrop, analyze or alter the data exchanged. Terms like information security, data confidentiality and integrity, entity authentication and identification need to be considered regarding the wireless setting [36]. The field of cryptography offers some solutions to the above issues but they need to be adapted suitably for their application on embedded devices.

As the model of ubiquitous computing arises, the use of low-end devices on a daily basis increases (e.g. access to a social network from a mobile phone). Thus, users will need a way to preserve their privacy and not reveal information that could be exploited by adversaries. For such issues, cryptography offers the tool of zero-knowledge proofs. A zero-knowledge proof can be used whenever someone needs to prove the possession of critical data without exchanging or revealing the actual data. Examples of today's Internet applications that use zero-knowledge proofs are e-commerce, e-voting, access authorization and entity authentication.

There exist various kinds of zero-knowledge proofs that involve problems like graph isomorphism and integer factorization. In this paper, we focus on the study of well-established zero-knowledge protocols based on the discrete logarithm problem. Up to now, although a wide variety of zero-knowledge protocols of this category has been proposed, (e.g. see [33]) no actual implementations regarding resource constrained devices have been presented. For this reason and regarding authentication and privacy issues on the Internet of Things, we concentrate on the application of zero-knowledge protocols for the security and privacy empowerment of current

[*]This work has been partially supported by the European Union under contract number ICT-2010-258885 (SPITFIRE).

wireless networks consisting of low constrained devices.

Nevertheless, when considering low-end devices capable of communicating wirelessly one should take into account the resource limitations, i.e. the restricted processing power and memory as well as the particularities imposed by the low-power wireless communication protocols (e.g. packet loss, channel throughput, message size). One can realize that high computation as well as high communication overhead leads to great energy consumption which is another very important constraint. These limitations consist the actual challenge, when trying to implement heavy protocols in terms of computation and communication, like zero-knowledge proofs, on constrained devices. Thus, in this work we emphasize on the elliptic curve cryptography approach, as proposed in [4], with the implementation of zero-knowledge protocols on constrained devices as our main objective.

1.1 Our Contributions

The contributions of this work are threefold.

Firstly, we study well-established zero-knowledge protocols based on the discrete logarithm problem (DLP) and we show how these protocols can be transformed and adapted under the elliptic curve discrete logarithm problem (ECDLP) and we state why these transformations are correct. This transformation step required a careful examination of the logarithmic exponential operations and the corresponding scalar multiplication on the elliptic curves as well as the arithmetic operations on the respective number fields. Such an adaptation consists the key for implementing zero-knowledge protocols on embedded devices. That is because elliptic curve cryptography (ECC) offers the same level of security with other public key cryptosystems (e.g. RSA) with the use of much smaller keys (see **Appendix**). This advantage ensures that we save space on the limited memory of such tiny devices, that the protocols' compiled code can actually fit well on them and that the protocols' message sizes are reasonable.

Secondly, we implement the new proposed protocols on Wiselib, a generic and open source algorithmic library. This way our code is generic, highly portable, publically available and ready to be used by developers that wish to provide certain levels of security and privacy in their applications.

Finally, we present a thorough evaluation of the new zero-knowledge protocols on two popular hardware platforms equipped with widely used low-end microcontrollers (Jennic JN5139, TI MSP430) as well as 802.15.4 [2] RF transceivers in terms of execution time, code size, messages' size and energy consumption. We experimentally prove that our protocols have small code footprint (around 8Kb) and that their exchanged messages fit well in the technical specifications of the 802.15.4 protocol. To the best of our knowledge, this is the first attempt of implementing and evaluating zero-knowledge protocols, with emphasis on low-constrained devices. Moreover, the resulting library can form the basis for the implementation of more complex protocols employed in various cryptographic applications, like attribute based credentials [8].

1.2 Paper Outline

The remaining of our paper is structured as follows: Firstly, in **Section 2** we present an overview of zero-knowledge proofs and in **Section 3** we show how well established zero knowledge protocols based on the discret logarithm problem can be adapted on the elliptic curve discrete logarithm problem. In **Section 4** we refer to the Wiselib platform on which we implemented the protocols, the hardware used for our experiments and we present an evaluation of the protocols on actual constrained devices. In **Section 5** we describe three everyday Internet of Things applications where zero-knowledge proofs can be used. In **Section 6** we conclude and propose some of our ideas for future work. Finally, for a reader's information in the **Appendix** we refer to the basic definitions of elliptic curve cryptography and the reasons that make it suitable for constrained devices.

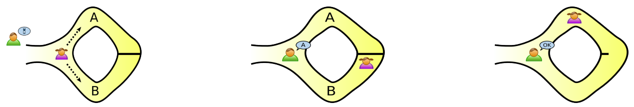
2. AN OVERVIEW OF ZERO KNOWLEDGE PROTOCOLS

Generally, a zero-knowledge protocol allows a proof of the truth of an assertion, while conveying no information whatsoever about the assertion itself other than its actual truth [26]. Usually, such a protocol involves two entities, a prover and a verifier. A zero-knowledge proof allows the prover to demonstrate knowledge of a secret while revealing no information whatsoever of use to the verifier in conveying this demonstration of knowledge to others.

The zero-knowledge protocols to be discussed are instances of interactive proof systems and non-interactive proof systems. In the first category, a prover and a verifier exchange multiple messages (challenges and responses), typically dependent on random numbers which they may keep secret whereas in the second the prover sends only one message. In both systems the prover's objective is to convince the verifier about the truth of an assertion, e.g. the claimed knowledge of a secret. The verifier either accepts or rejects the proof.

A zero-knowledge proof must obey the properties of completeness and soundness. A proof is **complete**, if given an honest prover and an honest verifier, the protocol succeeds with overwhelming probability and **sound** if the probability of a dishonest prover to complete the proof successfully is negligible [4]. Additionally, a protocol which consists a proof of knowledge must have the **zero-knowledge property**: there exists an expected polynomial-time algorithm which can produce, upon input of the assertions to be proven - but without interacting with the real prover, transcripts indistinguishable from those resulting from interaction with the real prover.

A typical example of zero-knowledge proof is known as Alibaba's cave problem [27]. In this story, Peggy has uncovered the secret word used to open a *magic* door in a cave. The cave is shaped like a circle, with the entrance on one side and the *magic* door blocking the opposite side, as shown in Figure 1. The left path from the entrance is labeled A and the right B. Victor states that he will pay her for the secret, but not until he's assured that she really knows it. Peggy claims that she will tell him the secret, but not until she receives the money. Thus, they devise a scheme by which Peggy can prove that she knows the *magic* word without telling it to Victor. The scheme steps are now described:



(a) 1st Step. (b) 2nd Step. (c) 3rd Step.

Figure 1: Alibaba's Cave Problem.

- Victor waits outside the cave as Peggy goes in
- Peggy randomly takes either path A or B inside the cave
- Victor enters the cave and shouts the name of the path he wants her to use to return either A or B, chosen at random
- Peggy does that using the secret word if needed to open the magic door
- The above steps are repeated n times until Victor is convinced that Peggy knows the secret word

Now, suppose that Peggy does not know the secret word. Since Victor chooses path A or B at random, Peggy has a 1/2 chance of cheating at one round. If the above steps are repeated for many rounds, Peggy's chance of successfully anticipating all of Victor's requests would become vanishingly small. Thus, if Peggy reliably appears at the exit Victor names, he can conclude that she is very likely to know the secret word.

Other problems that involve zero-knowledge proofs are the square root of an integer modulo n, graph isomorphism, integer factorization and the discrete logarithm problem. On this paper we focus on zero-knowledge protocols based on the discrete logarithm problem.

3. ZERO KNOWLEDGE PROTOCOLS BASED ON THE ECDLP

A wide variety of zero-knowledge protocols based on the Discrete Logarithm Problem (DLP) has been proposed so far, e.g. in [29], [33]. The Discrete Logarithm Problem is defined over arbitrary cyclic groups. A common example of cyclic group is the multiplicative group Z_n^* of order n, where n is a prime number and the group operation is multiplication modulo n. In such a group the Discrete Logarithm Problem (DLP) can be defined as follows: Given a prime n, a generator g of Z_n^* and an element $b \in Z_n^*$, find the integer x, $0 \leq x \leq n-2$ such that $g^x = b(mod\, n)$ [26].

Another common example of cyclic groups are elliptic curve groups which are defined over an additive group F of order n (note that n is no longer necessarily a prime number). The analogous problem to DLP over elliptic curve groups is called ECDLP (Elliptic Curve Discrete Logarithm Problem) and can be defined as follows: Given an elliptic curve E over a field F of order n (refered to as F_n from now on), a generator point $G \in E/F_n$ and a point $B \in E/F_n$ it is computationally hard to find x such that $B = x \cdot G$.

In this section, we show how well established zero-knowledge protocols based on the DLP can be adapted under the Elliptic Curve Discrete Logarithm Problem (ECDLP). This adaptation is a key step for porting such protocols to low constrained devices because of the Elliptic Curve Cryptography (ECC) advantages. As one can see in the **Appendix**, ECC can offer the same level of security as other public key cryptosystems, using smaller key sizes. This fact makes it suitable for implementations that concern constrained environments as it saves computational time and memory space and consequently reduces energy requirements. Such restrictions consist the real challenges when considering implementations on embedded devices.

3.1 Zero Knowledge Proof of Discrete Logarithm with Coin Flip

One of the first zero-knowledge protocols of discrete logarithm that was originally presented in [12]. Its elliptic curve analogous is as follows: Given an elliptic curve E over a field F_n, a generator point $G \in E/F_n$ and $B \in E/F_n$ Prover wants to prove that he knows x such that $B = x \cdot G$, without revealing x.

Protocol Steps:

- Prover generates random $r \in F_n$ and computes the point $A = r \cdot G$
- Prover sends the point A to Verifier
- Verifier flips a coin and informs the Prover about the outcome
- In case of HEADS Prover sends r to Verifier who checks that $r \cdot G = A$
- In case of TAILS Prover sends $m = x + r(mod\, n)$ to Verifier who checks that $m \cdot G = (x + r) \cdot G = x \cdot G + r \cdot G = A + B$

The above steps are repeated until Verifier is convinced that Prover knows x with probability $1 - 2^{-k}$ for k iterations.

Why it works: The protocol works as expected because in each iteration the steps to be executed depend on the outcome of the coin that the Verifier flips and the Prover cannot affect this. It needs to be executed for many iterations in order for the Prover's cheating probability to become very small. A dishonest Prover in each iteration can be prepared for only one of the coin outcomes and thus his cheating probability is 1/2. For example, if he prepares for TAILS he can generate a random m, compute $A = m \cdot G - B$ and send this point A to Verifier. But if HEADS come up this attack will not work. That is because he will need to compute a value $r \in F_n$ that generates A and that is an instance of the ECDLP. Thus, after k iterations, the Verifier is convinced with high probability $(1 - 2^{-k})$ that the Prover is honest.

3.2 Schnorr's Protocol

An improvement of the previous protocol was originally presented in [29]. The elliptic curve version of Schnorr's protocol, slightly modified, is the following: Prover and Verifier agree on an elliptic curve E over a field F_n, a generator $G \in E/F_n$. They both know $B \in E/F_n$ and Prover claims he knows x such that $B = x \cdot G$. He wants to prove this fact to Verifier without revealing x.

Protocol Steps:

- Prover generates random $r \in F_n$ and computes the point $A = r \cdot G$
- Prover sends the point A to Verifier
- Verifier computes random $c = HASH(G, B, A)$ and sends c to Prover
- Prover computes $m = r + c \cdot x (mod\, n)$ and sends m to Verifier
- Verifier checks that $P = m \cdot G - c \cdot B = (r + c \cdot x) \cdot G - c \cdot B = r \cdot G + c \cdot x \cdot G - c \cdot x \cdot G = r \cdot G = A$

Why it works: This protocol is superior to the previous one as it needs to be executed for one round. Verifier's coin flips (in correspondence with the Coin Flip protocol) are simulated using a hash function known only to him. A dishonest Prover has a tiny chance of cheating as he would have to fix the value of $P = m \cdot G - c \cdot B$ before receiving Verifier's hash value c. Under the assumption that the hash function used by the Verifier is secure, a Prover who does not know x, the discrete logarithm of B, cannot cheat.

3.3 Transforming Schnorr's Protocol to Digital Signature

In [18], the authors propose that with the use of a hash function and an agreement on an initial message m one can remove the interactivity from such protocols. The Verifier's random choices can be replaced with bits produced by a secure hash function. Thus, the next protocol is proposed.

Prover and Verifier agree on an elliptic curve E over a field F_n, a generator $G \in E/F_n$, a point $P \in E/F_n$ that represents the message the Prover wants to send and a hash function HASH (e.g. SHA-1). They both know $B \in E/F_n$. The Prover claims that he knows x such that $B = x \cdot G$ and he wishes to prove this fact to Verifier without revealing x.

Protocol Steps:

- Prover generates random $r \in F_n$ and computes the point $A = r \cdot G$
- Prover computes $c = HASH(x \cdot P, r \cdot P, r \cdot G)$
- Prover computes $s = r + c \cdot x (mod\, n)$
- Prover sends to Verifier the message: "$s || x \cdot P || r \cdot P || r \cdot G$"
- Verifier computes $c = HASH(x \cdot P, r \cdot P, r \cdot G)$
- Verifier checks that $s \cdot G = (r + c \cdot x) \cdot G = r \cdot G + c \cdot x \cdot G = r \cdot G + c \cdot B = A + c \cdot B$
- Verifier checks that $s \cdot P = (r + c \cdot x) \cdot P = r \cdot P + c \cdot xP$

Why it works: In this protocol we apply the non interactiveness trick proposed in [18]. The Prover simulates both the Prover and the Verifier with the use of a hash function and publishes the transcript of this whole dialogue. This way the Prover sends only one message and the Verifier either accepts or rejects. The Prover generates a random number as in previous protocols but the Verifier's random choices are simulated by hashing the input along with a value calculated from the Prover's choice of r. Thus, the Verifier's random choice depends on Prover's random choice and it is made hard to fake the outcome. The value c is really a challenge for the Prover as it is computed from the hash function and it is out of his control. If the Prover does not know x, in order to cheat he would try to find s satisfying $s \cdot G = r \cdot G + c \cdot x \cdot G$ which is an instance of the discrete logarithm problem. He could not cheat by enumerating random r values, as it would be too hard to find a matching value for c.

3.4 Zero Knowledge Test of Discrete Logarithm Equality

Suppose that Prover knows two publically known quantities that have the same discrete logarithm x to publicly known respective bases G and H of the group F_n.

Prover and Verifier agree on an elliptic curve E over a field F_n, a generator $G \in E/F_n$ and $H \in E/F_n$. Prover claims he knows x such that $B = x \cdot G$ and $C = x \cdot H$ and wants to prove knowledge of this fact without revaling x. The procedure was originally proposed in [7], and its ECC analogous is as follows:

Protocol Steps:

- Prover chooses random $r \in F_n$ and computes the points $K = r \cdot G$ and $L = r \cdot H$
- Prover sends the points K, L to Verifier
- Verifier chooses random $c \in F_n$ and sends c to Prover
- Prover computes $m = r + c \cdot x (mod\, n)$ and sends m to Verifier
- Verifier checks that $m \cdot G = (r + c \cdot x) \cdot G = r \cdot G + c \cdot x \cdot G = K + c \cdot B$
- Verifier checks that $m \cdot H = (r + c \cdot x) \cdot H = r \cdot H + c \cdot x \cdot H = L + c \cdot C$

Why it works: In this protocol the Prover claims he knows x as the discrete logarithm of two public quantities B, C. His actions are similar with Schnorr's protocol but for the two public quantities. For example in the first step he computes 2 points on the curve K, L that will be used for the verification. It can also be made non-interactive by the applying the Fiat-Shamir trick: the Prover simulates the Verifier by computing c with a secure hash function as HASH(B,G,C,H,K,L).

3.5 Zero Knowledge Proof of Single Bit

Prover and Verifier agree on an elliptic curve E over a field F_n, a generator $G \in E/F_n$ and $H \in E/F_n$. Prover knows x and h such that $B = x \cdot G + h \cdot H$ where $h = \pm 1$. He wishes to convince Verifier that he really does know x and that h really is ±1 without revealing x nor the sign bit [33].

Protocol Steps:

- Prover generates random $s, d, w \in F_n$
- Prover computes the points $A = s \cdot G - d \cdot (B + h \cdot H)$ and $C = w \cdot G$
- If $h = -1$ Prover swaps $A \leftrightarrow C$
- Prover sends the points A, C to Verifier
- Verifier generates random $c \in F_n$ and sends c to Prover
- Prover computes $e = c - d$ and $t = w + x \cdot e$ both $(mod\, n)$
- If $h = -1$ Prover swaps $d \leftrightarrow e$ and $s \leftrightarrow t$
- Prover sends to Verifier d, e, s, t
- Verifier checks that $e + d = c$, $s \cdot G = A + d \cdot (B + H)$ and that $t \cdot G = C + e \cdot (B - H)$

Why it works: It is straightforward to confirm that if B is really given by one of the two formulas the Prover claimed then Verifier's verification will succeed. It is also easy to see that the Prover does not give away any information that would allow the Verifier to deduce x nor the sign bit h. That is because x is hidden inside t after being multiplied with e and added in w. The sign bit h is randomized with the appropriate swaps in the case of -1.

4. PROTOCOLS IMPLEMENTATION AND EVALUATION

We implemented the new proposed zero-knowledge protocols based on the ECDLP using the Wiselib platform which is a generic algorithm library. Then, we evaluated the implemented protocols on two popular hardware platforms equipped with popular low-end microcontrollers (Jennic JN5139, TI MSP430) as well as 802.15.4 RF transceivers in terms of execution time, code size, message size and energy consumption.

4.1 Wiselib: A Generic Algorithm Library for Sensor Networks

We decided to implement our algorithms using **Wiselib** [5]: a code library, that allows implementations to be OS-independent. It is implemented based on C++ and templates, but without virtual inheritance and exceptions. Algorithm implementations can be recompiled for several platforms and firmwares, without the need to change the code. **Wiselib** can interface with systems implemented using C (Contiki), C++ (iSense), and nesC (TinyOS). A future plan for this library is to be adapted for C-based mobile phone operating systems like Android and iPhone OS.

Furthermore, an important feature of **Wiselib** are the already implemented algorithms and data structures. Since different kind of hardware uses different ways to store data (due to memory alignment, inability to support dynamic memory, etc.), it is important to use these safe types as much as possible since they have been tested before on most hardware platforms. As of mid 2010, the **Wiselib** includes about 40 Open Source implementations of standard algorithms, and is scheduled to grow to 150-200 algorithms by the end of 2011.

Additionally, **Wiselib** runs on the simulators **Shawn** [22] and **Tossim** [23], hereby easing the transition from simulation to actual devices. **Tossim** is a popular tool in the TinyOS community as it allows to simulate the exact source code that will run on the hardware and by using **Power-Tossim** [31] it can provide accurate estimates on the power consumption of an application. **Shawn** allows repeatability of simulations in an easy way by using only a single configuration file. It provides many options such as packet loss, radius of communication, ways of communicating and even mobility in an abstract way, without needing to provide specific code for every range. This **Wiselib** feature allows us to validate the faithfulness of our implementation and also get results concerning the quality of our algorithms without time consuming deployment procedures and harsh debugging environments.

Finally, an advantage of Wiselib is that with the aid of template specializations the algorithm code can be optimized and adapted for certain platforms. Depending on the compilation process, the compiler can select the code that fits best for the current platform (e.g. if there is a 32-bit processor) and exploit the presence of special platform hardware (e.g. the Jennic AES hardware for speedup of crypto routines).

4.2 Hardware

As mentioned earlier we used two different low-end devices for evaluating the implemented protocols. The first device is the Coalesenses iSense [9], [14] and the second is the Crossbow TelosB [1]. These devices are quite popular for their application in the area of wireless sensor networks.

The first device (iSense) consists of a Jennic JN5139 32-bit RISC controller [20] running at 16MHz. The ROM of this controller is 192Kb and its RAM is 96Kb that can be shared among program and data. It is equipped with 2.4Ghz IEEE 802.15.4 compliant RF transceiver (CC2420 chip) that can achieve bandwith up to 250 Kbps. Finally, this device runs the iSense firmware.

The second device (TelosB) consists of a Texas Instruments MSP430 [16] 16-bit microcontroller running at 8MHz. Its RAM is 10Kb and the program flash memory is 48Kb. This device is also equipped with 2.4GHz IEEE 802.15.4 compliant RF transceiver (CC2420 chip) able to achieve data rates up to 250Kbps. Finally, the TelosB device can run TinyOs 1.1.10 [34] (or higher) or the Contiky operating system [15].

4.3 Results

For the basic operations of Elliptic Curve Cryptography we have ported the implementation of [25] on the Wiselib. This implementation defines a recommended elliptic curve [30] over binary fields with equation $y^2 + xy = x^3 + x^2 + 1$ along with the irreducible polynomial $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$. The curve's order (the number of points on it) is

(a) A TelosB device.

(b) An iSense device.

Figure 2: The hardware used for our experiments.

$r = 0x40000000000000000000020108a2e0cc0d99f8a5ef$ and the base point is $G(x,y)$ where
$x = 0x2fe13c0537bbc11acaa07d793de4e6d5e5c94eee8$ and
$y = 0x289070fb05d38ff58321f2e800536d538ccdaa3d9$. The execution time of the basic elliptic curve operations for both the microprocessors used can be seen on Table 1. We note that there has been no attempt to optimize the code responsible for the elliptic curve operations.

Operation	Execution Time	
	JN5139	MSP430
Private Key Generation	0.087 sec	0.3 sec
Public Key Generation (scalar multiplication)	11.121 sec	58.02 sec
Curve Points Addition	0.094 sec	0.29 sec

Table 1: Execution Time of Basic Elliptic Curve Operations on the JN5139 and MSP430 microprocessors.

Table 2 summarizes the execution times of some additional arithmetic operations used by the protocols on the JN5139 and MSP430 microcontrollers. As a hash function for the protocols that require one we used the algorithm SHA-1 [17].

Operation	Execution Time	
	JN5139	MSP430
Private Key Addition (21 bytes)	0.005 sec	0.012 sec
Private Key Multiplication (21 bytes)	0.014 sec	0.02 sec
SHA-1 Hash (250 bytes)	0.02 sec	0.031 sec

Table 2: Execution Time of Arithmetic Operations Used by the Protocols on the JN5139 and MSP430 microprocessors.

From these experiments, a reader can observe that the elliptic curve scalar multiplication (Public Key Generation) is the most demanding arithmetic operation. Moreover, it is also evident that the JN5139 (16MHz) calculates much faster all the operations examined than the MSP430 (8MHz). We believe that a delay of 11 sec for the generation of a public key is acceptable. On the other hand, the 58 sec that are required from the MSP430 is probably too long.

For all the arithmetic operations previously mentioned, we present a theoretical approach for estimating their energy consumption. This approach is based on the current consumption that the data sheet of each microprocessor provides us. For example, from the graphs presented in JN5139 data sheet we get that the current consumption of the mote when the microprocessor works on maximum load is 12.7mA, on input voltage 3V and temperature 25°C. The input voltage 3V is a reasonable choice considering the fact that two AA batteries provide that much voltage. Respectively, the MSP430 microcontroller draws approximately 1.8mA current when it operates. Thus, considering the formula

$$E = V \cdot I \cdot t \tag{1}$$

where V is voltage, I is current and t is time, and the execution time of each operation, as seen on Tables 1 and 2, we show on Table 3 our estimation for each operation's energy consumption.

Operation	Energy Consumption	
	JN5139	MSP430
Private Key Generation	3.31 mJ	1.62 mJ
Public Key Generation (scalar multiplication)	423.6 mJ	313.3 mJ
Curve Points Addition	3.42 mJ	1.56 mJ
Private Key Addition (21 bytes)	0.19 mJ	0.06 mJ
Private Key Multiplication (21 bytes)	0.53 mJ	0.1 mJ
SHA-1 Hash (250 bytes)	0.76 mJ	0.16 mJ

Table 3: Energy Consumption of Basic Arithmetic Operations Used by the Protocols on the JN5139 and MSP430 microprocessors.

As expected, on Table 3, we observe that the elliptic curve scalar multiplication is the most energy consuming arithmetic operation. Additionally, it is interesting to note that the JN5139 processor, although it is much faster in computation than the MSP430, it is less efficient in terms of energy consumption.

Next, we have concentrated the prover's (PRV) and verifier's (VER) actions for each of the implemented protocols. Table 4 summons up these actions. Regarding to these actions we verify the total protocol execution times that can be seen subsequently.

On Table 5 one can see the total execution time of the above implemented protocols. This is measured as the time space between the prover's beginning of the protocol until the verifier's final response of whether he accepts or rejects the proof. We can observe that an interactive protocol like the first one, which requires a large number of execution rounds for verification, is not suitable for low-constrained devices. Thus, the usage of protocols (like the rest of them) that need one round of execution is advised [3].

Table 6 describes the messages and their sizes, exchanged by the prover (PRV) and verifier (VER) for the completion of each protocol. Message size is an important parameter

Protocol / Operation	Random Key Generation	Curve Multiplication	Curve Addition	SHA-1 Hash	Private Keys Addition	Private Keys Multiplication	Msgs Sent
ZKP of DL with Coin Flip PRV	1	1	-	-	1 (if tails)	-	2
ZKP of DL with Coin Flip VER	-	1	1 (if tails)	-	-	-	2
Schnorr's Protocol PRV	1	1	-	-	1	1	2
Schnorr's Protocol VER	-	2	1	2	-	-	2
Schnorr's Signature Protocol PRV	1	3	-	1	1	1	1
Schnorr's Signature Protocol VER	-	4	2	1	-	-	1
ZKP of DL Equality PRV	1	2	-	-	1	1	2
ZKP of DL Equality VER	1	4	2	-	-	-	2
ZKP of Single Bit PRV	3	3	2	-	2	1	2
ZKP of Single Bit VER	1	4	4	-	1	-	2

Table 4: Actions Held by the Prover (PRV) and Verifier (VER) for each protocol.

Protocol	Required Rounds	Total Execution Time	
		iSense	TelosB
ZKP of DL with Coin Flip	100 or more	2277 sec	11802 sec
Schnorr's Protocol	1	33.894 sec	172.77 sec
Schnorr's Signature Protocol	1	78.645 sec	396.57 sec
ZKP of DL Equality	1	68.596 sec	346.2 sec
ZKP of Single Bit	1	80.46 sec	462.74 sec

Table 5: Total Execution Time of the Protocols on the Devices Used.

when considering low power wireless communication protocols like 802.15.4. All of our protocols messages except Schnorr's Non-Interactive Protocol PRV message (149 bytes which had to be broken in two pieces), fit well on a single 802.15.4 packet (128 bytes). This fact is advantageous as bigger messages would result to more message exchanges which in turn would result to greater energy consumption and possibly to wireless medium congestion.

Protocol	Message Size
ZKP of DL with Coin Flip PRV	Point Message: 43 bytes New Key Message: 22 bytes
ZKP of DL with Coin Flip VER	Coin Message: 2 bytes Final Message: 1 byte
Schnorr's Protocol PRV	Point Message: 43 bytes New Key Message: 22 bytes
Schnorr's Protocol VER	Hash Message: 22 bytes Final Message: 1 byte
Schnorr's Signature Protocol PRV	Point and Key Message: 149 bytes (in two pieces)
Schnorr's Signature Protocol VER	Final Message: 1 byte
ZKP of DL Equality PRV	Points Message: 85 bytes New Key Message: 22 bytes
ZKP of DL Equality VER	New Key Message: 22 bytes Final Message: 1 byte
ZKP of Single Bit PRV	Points Message: 85 bytes New Key Message: 85 bytes
ZKP of Single Bit VER	New Key Message: 22 bytes Final Message: 1 byte

Table 6: Size of Messages Exchanged by the Prover (PRV) and the Verifier (VER) for each Protocol.

It is quite difficult to estimate the energy consumption of RF messages as they depend on various factors like motes distance, obstacles presence and weather conditions. However, we try to make a theoritical estimation about it. Under the *ideal* hypothesis that the devices' RF transceivers achieve data rate 250 Kbps and according to Table 6 we calculate the time required to send each message depending on its size (e.g. with data rate 250 Kbps a message of size 85 bytes needs around 2.72 msec for transmission/reception). Next, we estimate the energy consumption for sending and receiving the messages on each device using the formula 1. From the iSense data sheet we get that the Rx current consumption on input voltage 3V and temperature 25°C is 43.7 mA and the Tx current consumption is 39.9 mA. Respectively, from the TelosB data sheet we get that Rx current consumption is 23mA and that Tx current consumption is 21mA. Table 7 summarizes the results.

Message Size	iSense		TelosB	
	Tx	Rx	Tx	Rx
149 bytes	0.5 mJ	0.6 mJ	0.3 mJ	0.32 mJ
85 bytes	0.32 mJ	0.36 mJ	0.17 mJ	0.18 mJ
43 bytes	0.16 mJ	0.18 mJ	0.08 mJ	0.09 mJ
22 bytes	0.08 mJ	0.09 mJ	0.045 mJ	0.048 mJ
2 bytes	0.007 mJ	0.008 mJ	0.004 mJ	0.004 mJ
1 byte	0.003 mJ	0.004 mJ	0.002 mJ	0.002 mJ

Table 7: Message Tx, Rx Energy Consumption According to its Size on the Devices Used.

In correspondence with the Tables 3, 4, 7 we estimate the total energy consumption for the prover and verifier of each protocol depending on its actions (arithmetic operations and messages sent/received). Table 8 shows the results. Once again, we realize why a protocol (like ZKP of DL with Coin Flip) that requires many execution rounds, is not suitable for applications that involve constrained devices. Moreover, we observe that the protocols execution consumes less energy (although more time) on the TelosB device with the slower processor.

From tables 5 and 8, we note that Schnorr's protocol is the fastest one and thus consumes the least energy. We think

Protocol	Total Energy Consumption	
	iSense	TelosB
ZKP of DL with Coin Flip PRV	42.7 J	31.5 J
ZKP of DL with Coin Flip VER	42.5 J	31.4 J
Schnorr's Protocol PRV	0.42 J	0.31 J
Schnorr's Protocol VER	0.85 J	0.62 J
Schnorr's Signature Protocol PRV	1.27 J	0.94 J
Schnorr's Signature Protocol VER	1.7 J	1.25 J
ZKP of DL Equality PRV	0.85 J	0.62 J
ZKP of DL Equality VER	1.7 J	1.25 J
ZKP of Single Bit PRV	1.28 J	0.94 J
ZKP of Single Bit VER	1.71 J	1.26 J

Table 8: Total Energy Consumption for the Prover (PRV) and the Verifier (VER) of each Protocol on the Devices Used.

that 33 sec for the completion of a zero-knowledge proof on a device with limited processing power is quite fair.

As a last experiment, we measured the code size of the above protocols on the devices used. The compiled code actually fits fairly well (approximately 8Kb) on the tiny memory of their processors. The results can be seen on Table 9. The difference in the protocols code size on the two devices is reasonable as different compilers (ba-elf for iSense and mspgcc for TelosB) are employed.

Protocol	Total Code Size (text + data + bss)	
	iSense	TelosB
ZKP of DL with Coin Flip PRV	7908 bytes	6517 bytes
ZKP of DL with Coin Flip VER	7004 bytes	6289 bytes
Schnorr's Protocol PRV	7964 bytes	6759 bytes
Schnorr's Protocol VER	9292 bytes	9455 bytes
Schnorr's Signature Protocol PRV	10584 bytes	10433 bytes
Schnorr's Signature Protocol VER	9540 bytes	10053 bytes
ZKP of DL Equality PRV	8568 bytes	7697 bytes
ZKP of DL Equality VER	8964 bytes	8519 bytes
ZKP of Single Bit PRV	9604 bytes	9471 bytes
ZKP of Single Bit VER	9032 bytes	7455 bytes

Table 9: Code Size of the Protocols for Prover the (PRV) and the Verifier (VER) on the Devices Used.

4.4 Discussion

From the results presented in the previous subsection we stick to the most important observations.

First of all, we state why implementing the ZKP protocols under the elliptic curve cryptography setting is advantageous. As presented in the **Appendix**, ECC offers the same level of security as other public key cryptosystems, using smaller key sizes. An implementation of zero knowledge protocols of discrete logarithm over multiplicative groups would require the use of at least 1024-bit keys as proposed by NIST (see Figure 5 in Appendix). A reader can realize that using elliptic curve groups instead of multiplicative groups, the arithmetic operations need less time to execute, less memory space and thus less energy consumption. In fact, it has been proven that ECC actually outperforms RSA on constrained environments in terms of computation time, memory requirements and thus energy consumption [19]. Moreover, considering the wireless nature of communication on the embedded setting ECC offers the advantage of smaller message sizes that cost less and have better chances of being delivered.

Concerning the protocols performance we discuss the experimental results.

The first proposed protocol (ZKP of DL with Coin Flip) requires a large number of execution rounds in order to successfully complete. It performs a large number of arithmetic operations and it involves many message exchanges. Although such a protocol could be executed fairly in non-embedded systems, when considering embedded devices capable of wireless communication it is not suitable. It is very demanding in terms of energy consumption and the messages exchanged can congest the wireless medium. The rest of the protocols require only one round of execution [3], much smaller number of messages (2-4 messages) and complete much faster and with less energy consumption. Schnorr's protocol required the least time to execute - 33 sec on the JN5139 microcontroller, which is quite fair considering the processing power and the memory of the device used.

An additional disadvantage of the first protocol is that it cannot be considered as secure as the rest when dealing with wireless communication. A malicious verifier or an adversary who eavesdrops the communication between the prover and the verifier can replay the proof to another party using the overheard data. That is because the verifier's responses consist of just a single bit (simulating a coin flip). In all the other protocols the Verifier responses involve fresh random data (e.g. using a hash function) and such an attack could not work.

Another issue concerns the comparison between Schnorr's and Schnorr's non-interactive protocol. By transforming Schnorr's protocol to a non-interactive protocol, the required message exchanges are reduced (1 message required by the prover and verifier). However, the transformed protocol is not as efficient as Schnorr's original protocol in terms of execution time, energy consumption, code size and message size (see below).

As far as the efficiency of the elliptic curve arithmetic operations of our implementation is concerned, we have already mentioned that we did not make any attempts to optimize the code. One can easily observe that the curve point multiplication which is the basic action for each protocol, needs the most time to execute (11 sec on JN5139, 58 sec on MSP430). Subsequently, this action requires the most energy for its execution. We believe that this is the *price* you pay when you write some generic code which can be portable. Although, Wiselib offers the chance to compile your code exploiting some platform features (a 32-bit processor or hardware speedups) we aimed at generality and portability. The goal of this work was not to achieve some faster platform specific code (e.g. by employing low-level assembly code) as in [19], [24].

However, we think that for such low-end microprocessors running at 16 MHz or 8MHz the total execution time of our protocols (except of course ZKP of DL with Coin Flip) is reasonable. For example, 33 sec on a 16MHz microcontroller for a secure verification is not too much to wait. Having implemented our protocols in Wiselib, with little effort we can port our code on C-based operating systems for mobile phones like Android and iPhone OS where the embedded hardware is much more powerful. For instance, we note that a current HTC mobile phone is powered by an 1 GHz ARMv7 Snapdragon processor with 512 MB of flash and 576 MB of RAM. Of course, our code would execute much faster on such a platform. Nevertheless, running and evaluating it on really low-end devices was much more meaningful.

Considering memory limitations, we showed that the protocols' compiled code actually fits well on the restricted memory of the devices used. For the iSense devices the protocols code occupied approximately 9Kb out of the total 96Kb of memory that is offered. Respectively, the protocols code on the TelosB devices, occupied approximately 8Kb out of the total flash memory of 48Kb.

Moreover, as far as the protocols' message sizes are concerned, we observe that they are acceptable. Most of the messages exchanged are 43 or 85 bytes which can fit on a single 802.15.4 packet (max payload 128 bytes). Only, the Prover's message from Schnorr's non-interactive protocol was large enough (149 bytes) that could not fit in a single packet and thus had to be broken in two pieces. However, by recompiling each device's firmware, which is a straightforward procedure, one could increase the maximum payload size and such a message could fit in a single packet. Such a solution though, would be non-standard. The reasonable message sizes are due to the ECC approach. A reader can realize that using another cryptosystem (e.g. with 1024-bit keys) would result in larger message sizes that would require fragmentation. This way many messages would need to be exchanged and more energy would be consumed. Additionally, the possibility of communication faults would increase as the wireless medium could be congested.

Finally, a point for discussion, is the fact that in our protocols we preload the elliptic curve used and its parameters on the devices memory. This way, the elliptic curve and its parameters could be compromised by physical and tampering attacks. However, we believe that in the future, most devices will be equipped with Trusted Platform Module (TPM), secure cryptoprocessors suitable for storing cryptographic keys and protecting private information.

5. APPLICATION SCENARIOS

We are strongly confident that zero-knowledge proofs can be used as a privacy-preserving tool in future global networks consisting of different kinds of devices and thus we present some everyday's application examples:

5.1 Course Polling at University

In most universities of the industrial world, course evaluations have become standard practise but they are typically conducted outside computers in order to protect student's privacy. An application that would encourage students to anonymously state their opinion about a course is descirbed.

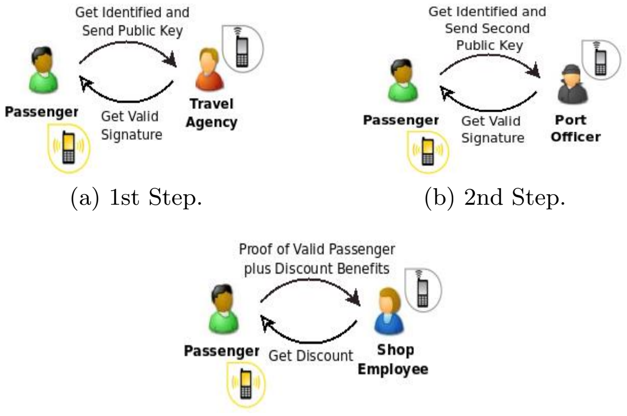
Imagine a concept where students can provide feedback about a course or a lecture using tiny devices capable of communicating wirelessly (e.g. clickers). For the success of this concept two parameters should be considered. Firstly, only authorized students (students who actually attend the course) should be able to poll and secondly students' anonymity should be guaranteed. Thus, when a student enrols for a course at the University's Administration Office (a trusted authority) he gets a tiny device that holds a private key encoding some of his attributes (e.g. name, matriculation number, course id and some fresh random value). This private key is used to generate a public key which is stored by the authority on a list. This list is then provided to the professor who is responsible for the course. At the end of a lecture the professor activates a polling application on his laptop (equipped with RF transceiver) that presents questions to the students. The students can now answer the poll questions using their devices. The application accepts a student's answer if the student's device presents a public key that exists on the list of valid keys (as provided by the University's Administration Office) and proves that it is the actual owner of the private key that has generated it. This way, student's anonymity is preserved (his private key is never revealed) and the professor gets the poll results while being assured that students who do not attend the class or malicious entities who eavesdrop the messages exchanged during the polling procedure are not able to poll.

For the deployment of such an application, a zero-knowledge protocol of discrete logarithm is required. Regarding the performance evaluation results that were presented in the previous section we propose *Schnorr's Protocol*.

5.2 Anonymous Travel Ticket plus Discount Benefits

Nowadays, it is quite common for passenger ships that travel over national waters to have duty free shops in them (e.g. boats that travel on the Baltic Sea). When passengers enter these shops, they are usually requested to present their ticket in order to buy some goods. As travel tickets typically contain a passenger's name, privacy issues are raised. For example, a passenger may not desire for his purchases to be publically known e.g. for statistical purposes. However, a passenger that has the benefit of some discount (e.g. a first class passenger) needs a way to prove this fact without revealing his private data. We present an application that can deal with these issues (Figure 3).

First Step: A passenger visits a travel agency office in order to buy a boat ticket. As he purchases the ticket, his mobile phone which is able to communicate wirelessly through Near Field Communications (NFC) stores a private key that en-



(a) 1st Step.

(b) 2nd Step.

(c) 3rd Step.

Figure 3: Boat Travel Ticket plus Discount Benefits Application.

codes his attributes (name, travel class and some random fresh value). This private key is used to generate a public key that is then digitally signed by the travel agency with a secret key depending on the passenger's travel class.

Second Step: When passengers approach the port entrance, they identify themeselves to the port officer as proposed by international travel safety regulations. Once a passenger that travels first class (and desires the right to some discount benefits on the ship shops) is identified by the port officer, he generates a second public key that encodes his right to discount with the same private key using his mobile phone. The second key is digitally signed by the port officer's device.

Third Step: From now on, a passenger is not required to identify himself at any occasion on the boat. When a passenger enters the ship his mobile phone communicates with the device of the ticket control employee and proves that he is eligible to travel at the corresponding class without revealing his private data (using a zero-knowledge proof of discrete logarithm e.g. *Schnorr's protocol*). When a first class passenger who has additionally discount benefits wants to purchase some products he enters the duty free shop and his mobile phone communicates through NFC with the employee's device and proves two things: first that he is a valid first class passenger and second that he has the right to product discount. This proof takes place by showing that two digitally signed by the corresponding authorities public keys have the same discrete logarithm (use of the protocol *Zero Knowledge Test of Discrete Logarithm Equality*). If the proof succeeds the passenger gets a discount on his purchases. A malicious user, is not able to extract information about passenger's private data (e.g. his name) or to get product discount if he does not match the necessary criteria.

5.3 Parking in Smart Cities

Finally, we show how the protocol *Zero-Knowledge Proof of Single Bit* can be used in an application. This protocol proves the possesion of a discrete logarithm without revealing it and additionally proves that a value has been added or substracted to the corresponding public key without revealing which operation took place.

As cities get more and more *intelligent*, the concept of smart parking is becoming reality. Ultra low power wireless mesh networks with web-based suites of parking management applications are deployed in order to provide real time parking occupancy status, charging fees and other useful information. A typical smart parking scenario is allowing citizens to park without charge at the spaces of their neigborhood. However, such an application could raise important privacy issues as an attacker could extract from the network private information about the citizens (e.g. Mr. Smith has just arrived home). We propose a scenario that protects citizen's privacy.

In this scenario, we assume that there exists a parking space between every two major streets inside a city. Each citizen gets an RFID tag containing a private key that encodes his attributes (name, vehicle registration plate and some fresh random value) from the local goverment. The private key is used to generate a public key that proves his validity as a citizen. Moreover, depending on his address, a value that encodes the specific neighborhood parking space (which should be free only for those who live on one of the two corresponding streets) is added to or substracted from his public key. The authority digitally signs the new public key and then places this RFID tag on the citizen's vehicle.

Parking spaces are equipped with RFID tag readers placed on their entrances. Thus, when a citizen wants to park in his neigborhood parking space, his device communicates with the reader on the parking entrance and proves two things: first that this vehicle belongs to a valid citizen and second that this citizen lives on one of the two neigborhood streets (without revealing which) and is able to park without charge. This proof is done by presenting a public key signed from the local goverment (the reader is aware of the trusted authority's public key), by proving that it actually holds the private key that generated the citizen validity public key and by proving that the specific parking value has been added to or substracted from the public key. If the proof is successful the parking entrance bar is lifted and the vehicle is allowed to enter the parking. Another citizen's vehicle that is not eligible to park in that space (the parking value has not been added or substracted) is requested to pay the appropriate fee in order to enter the parking space and a vehicle that is not registered at all to the local goverment is not allowed enter the parking. A malicious user who eavesdrops the proof cannot extract any information that could be useful to him (e.g. in order to get free parking space or monitor a citizen's actions).

6. CONCLUSIONS AND FUTURE WORK

In this paper, we considered the problem of implementing zero-knowledge protocols (ZKP) in low-end devices. Considering the resource limitations of such devices as well as the restrictions imposed by low power wireless communication protocols (e.g. IEEE 802.15.4) we applied the elliptic curve cryptography (ECC) approach. Specifically, we have carefully transformed well established zero-knowledge protocols based on the discrete logarithm problem (DLP) under the elliptic curve discrete logarithm problem (ECDLP).

This transformation step was the key for implementing such heavy protocols, in terms of computation and communication, on constrained environments, due to the fact that ECC offers similar levels of security with other cryptosystems (e.g. RSA) using smaller keys. For the first time, we present an implementation of ZKP protocols in an open source and generic programming library called Wiselib. Our code is highly portable, freely available and ready to use [35]. Based on our implementations, we conducted a thorough and comparative evaluation of the protocols on two popular hardware platforms equipped with widely used low-end microcontrollers (Jennic JN5139, TI MSP430) as well as 802.15.4 RF transceivers. We believe that our results can be used by developers that wish to provide certain levels of security and privacy in their applications.

Future work includes the expansion of the library presented in this paper so as to include ZKIP protocols which prove various relations for the encoded values without revealing them (e.g. prove that my age is over 18 years without revealing it). This way, our library can be the basis for implementing attribute based credentials [10], [8] on embedded devices.

7. REFERENCES

[1] Crossbow technology inc. http://www.xbow.com.

[2] Ieee standard 802.15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks lr-wpans, 2003.

[3] S. Almuhammadi and C. Neuman. Security and privacy using one-round zero-knowledge proofs. In *CEC*, pages 435–438, 2005.

[4] S. Almuhammadi, N. T. Sui, and D. McLeod. Better privacy and security in e-commerce: Using elliptic curve-based zero-knowledge proofs. In *CEC*, pages 299–302, 2004.

[5] T. Baumgartner, I. Chatzigiannakis, S. P. Fekete, C. Koninis, A. Kröller, and A. Pyrgelis. Wiselib: A generic algorithm library for heterogeneous sensor networks. In *EWSN*, pages 162–177, 2010.

[6] I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic curves in cryptography*. Cambridge University Press, New York, NY, USA, 1999.

[7] J. Boyar, D. Chaum, I. Damgård, and T. P. Pedersen. Convertible undeniable signatures. In A. Menezes and S. A. Vanstone, editors, *CRYPTO*, volume 537 of *Lecture Notes in Computer Science*, pages 189–205. Springer, 1990.

[8] S. A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000.

[9] C. Buschmann and D. Pfisterer. iSense: A modular hardware and software platform for wireless sensor networks. Technical report, 6. Fachgespräch Drahtlose Sensornetze der GI/ITG-Fachgruppe Kommunikation und Verteilte Systeme, 2007.

[10] J. Camenisch and E. V. Herreweghen. Design and implementation of the idemix anonymous credential system. In *ACM Conference on Computer and Communications Security*, pages 21–30, 2002.

[11] Certicom. An elliptic curve cryptography (ecc) primer: why ecc is the next generation of public key cryptography. 2004.

[12] D. Chaum, J.-H. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *EUROCRYPT*, pages 127–141, 1987.

[13] J. Chen and J. Wu. A survey on cryptography applied to secure mobile ad hoc networks and wireless sensor networks, 2010.

[14] coalsesenses GmbH. http://www.coaelesenses.com.

[15] Contiki, the operating system for connecting the next billion devices - the internet of things. http://www.sics.se/contiki/.

[16] J. H. Davies. *MSP430 Microcontroller Basics*. Newnes, Newton, MA, USA, 2008.

[17] D. Eastlake and P. Jones. US Secure Hash Algorithm 1 (SHA1). Technical Report 3174, Sept. 2001.

[18] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.

[19] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. pages 119–132. 2004.

[20] Jennic Ltd. Product Brief JN513x, IEEE802.15.4 and ZigBee Wireless Microcontrollers, 2007.

[21] N. Koblitz, A. J. Menezes, Y.-H. Wu, and R. J. Zuccherato. *Algebraic aspects of cryptography*. Springer-Verlag New York, Inc., New York, NY, USA, 1998.

[22] A. Kröller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer. Shawn: A new approach to simulating wireless sensor networks. *CoRR*, abs/cs/0502003, 2005.

[23] P. Levis, N. Lee, M. Welsh, and D. E. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys*, pages 126–137, 2003.

[24] A. Liu and P. Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *IPSN*, pages 245–256, 2008.

[25] D. J. Malan, M. Welsh, and M. D. Smith. Implementing public-key infrastructure for sensor networks. *TOSN*, 4(4), 2008.

[26] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[27] J.-J. Quisquater, M. Quisquater, M. Quisquater, M. Quisquater, L. C. Guillou, M. A. Guillou, G. Guillou, A. Guillou, G. Guillou, S. Guillou, and T. A. Berson. How to explain zero-knowledge protocols to your children. In *CRYPTO*, pages 628–631, 1989.

[28] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[29] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991. 10.1007/BF00196725.

[30] Certicom research : Sec 2 - recommended elliptic curve domain parameters. http://www.secg.org/collateral/sec2_final.pdf,

2010.

[31] V. Shnayder, M. Hempstead, B.-r. Chen, G. W. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 188–200, New York, NY, USA, 2004. ACM.

[32] J. H. Silverman. *The Arithmetic of Elliptic Curves, 2nd Edition*. Springer Verlag, 2009.

[33] W. Smith. Cryptography meets voting, 2005.

[34] TinyOS. http://www.tinyos.net.

[35] Wiselib : A generic algorithm library for sensor networks. http://wisebed.eu/wiselib/.

[36] Y. Zhou, Y. Fang, and Y. Zhang. Securing wireless sensor networks: a survey. *Communications Surveys Tutorials, IEEE*, 10(3):6 –28, 2008.

APPENDIX

A. APPENDIX

A.1 Elliptic Curve Cryptography

For a reader's information, in this section we review some basic concepts of elliptic curves and their definition over finite fields. Moreover, we discuss about the elliptic curve cryptography (ECC) approach and the reasons it consists the best choice for the implementation of asymmetric cryptography on constrained devices.

Elliptic curves are usually defined over real numbers (Figure 4), over a *binary field* $F_{2^m}(m \geq 1)$, or over a *prime field* $F_p, (p > 3)$. An elliptic curve E over a binary field F_{2^m}, where $m \geq 1$ consists of the set of points (x, y) that satisfy the equation

$$y^2 + xy = x^3 + ax^2 + b \tag{2}$$

The set of solutions (x, y) of Equation (2) along with a point O, called the point at infinity and a special addition operation form an elliptic curve group over F_{2^m}. The order m of an elliptic curve is the number of points on $E(F_{2^m})$. The dominant operation in ECC cryptographic schemes is point multiplication. This operation is the key for the use of elliptic curves for asymmetric cryptography - the critical operation which is itself fairly simple, but whose inverse (the elliptic curve discrete logarithm problem, see below) is computationally hard. The security of elliptic curve cryptosystems is based on the difficulty of solving the discrete logarithm problem on the elliptic curve group. The Elliptic Curve Discrete Logarithm Problem (ECDLP) is about determining the least positive integer k which satisfies the equation $Q = k \cdot P$ for two given points Q and P on the elliptic curve group. The fastest known algorithms for solving the ECDLP need exponential time in the worst case, while for solving the same problem on non-elliptic curve based groups, the fastest known algorithms need subexponential time [21]. For more information on elliptic curves a reader is advised to check on the definitive books for ECC, [6] and [32].

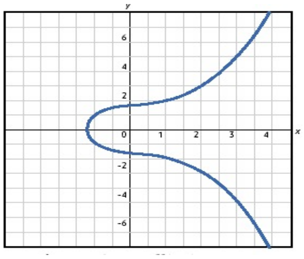
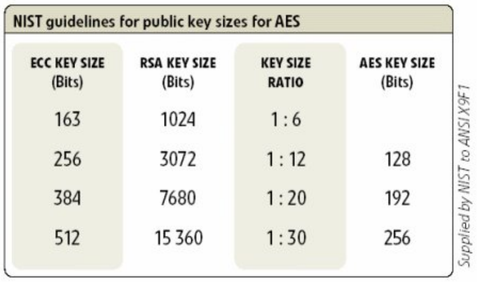


Figure 4: An Elliptic Curve over Real Numbers.

From the previous definition, one can realize that Elliptic Curve Cryptography (ECC), is a relative of discrete logarithm cryptography. What makes it the best choice for asymmetric cryptography in comparison with other public key cryptosystmes (e.g. RSA [28] which is based on the problem of factoring large numbers) is that it offers higher security at the same bits levels. This advantage exists because of the difference in the method by which a group is defined, how the elements of the group are defined and how the fundamental operations are held. These different definitions are what gives ECC its more rapid increase in security as key length increases.

Consequently, the reader can realize that the ECC inverse operation (solving the ECDLP) gets harder, faster, against increasing key length, in comparison with the inverse operation in RSA (solving the integer factorization problem). This means that as security requirements become more stringent, and as processing power gets more expensive, ECC becomes the more practical system for use. This keeps ECC implementations smaller and more efficient than other implementations. As a result, ECC can use a considerably shorter key and offer the same level of security, whereas other asymmetric algorithms using much larger ones. Moreover, the gulf between ECC and its competitors in terms of key size required for a given level of security becomes dramatically more pronounced, at higher levels of security [11]. In Figure 5, one can see the equivalent key sizes for ECC and RSA.



NIST guidelines for public key sizes for AES

ECC KEY SIZE (Bits)	RSA KEY SIZE (Bits)	KEY SIZE RATIO	AES KEY SIZE (Bits)
163	1024	1 : 6	
256	3072	1 : 12	128
384	7680	1 : 20	192
512	15 360	1 : 30	256

Supplied by NIST to ANSI X9F1

Figure 5: Equivalent Key Sizes and Key Ratio for ECC and RSA.

Conclusively, due to the arguments stated before, ECC is an excellent choice for doing asymmetric cryptography in portable, necessarily constrained devices. The smaller ECC keys mean that the cryptographic operations to be performed by the communicating devices can be squeezed into considerably smaller hardware, that the software applications may complete cryptographic operations with fewer processor cycles and that operations can be performed much faster, while still guaranteeing equivalent security. In turn, this means less heat and less power consumption on the devices' chips as well as software applications that run faster with lower memory demands.