Research Article

Adaptable Security in Wireless Sensor Networks by Using Reconfigurable ECC Hardware Coprocessors

J. Portilla,[1] A. Otero,[1] E. de la Torre,[1] T. Riesgo,[1] O. Stecklina,[2] S. Peter,[2] and P. Langendörfer[2]

[1] *Centro de Electronica Industrial, Universidad Politecnica de Madrid, Jose Gutierrez Abascal 2, 28006 Madrid, Spain*
[2] *IHP, Im Technologiepark 25, 15236 Frankfurt, Germany*

Correspondence should be addressed to J. Portilla, jorge.portilla@upm.es

Specific features of Wireless Sensor Networks (WSNs) like the open accessibility to nodes, or the easy observability of radio communications, lead to severe security challenges. The application of traditional security schemes on sensor nodes is limited due to the restricted computation capability, low-power availability, and the inherent low data rate. In order to avoid dependencies on a compromised level of security, a WSN node with a microcontroller and a Field Programmable Gate Array (FPGA) is used along this work to implement a state-of-the art solution based on ECC (Elliptic Curve Cryptography). In this paper it is described how the reconfiguration possibilities of the system can be used to adapt ECC parameters in order to increase or reduce the security level depending on the application scenario or the energy budget. Two setups have been created to compare the software- and hardware-supported approaches. According to the results, the FPGA-based ECC implementation requires three orders of magnitude less energy, compared with a low power microcontroller implementation, even considering the power consumption overhead introduced by the hardware reconfiguration.

1. Introduction

Wireless sensor networks represent one of the most outstanding technologies in the last years, and several research disciplines have appeared to cover the special needs of this kind of systems, like hardware platforms, communication protocols, security issues, or operating systems, among others.

A typical application scenario for WSNs is the deployment of a set of unattended nodes that operate autonomously in an environment, which is to be monitored. Sensor nodes are often deployed in areas where they are freely accessible. Thus their position, the hardware and even the software can be compromised. In order to ensure correct system behavior it is essential to protect wireless sensor nodes and the wireless communication. Without protection the network can be observed by villains, and confidential information can be eavesdropped. Encryption is a reasonable countermeasure to protect data, while it also adds new processing load to the nodes. Applying strong cryptographic means to ensure security of the system is often contradicting the requirement that the WSN need to run for several months or even years, without human intervention. Traditional

microcontroller-based sensor nodes do not provide sufficient computation power to process strong cryptomeans, and even if these operations are executed they consume a significant amount of energy depleting the battery very quickly.

Moreover, the nature of sensor networks makes their security requirements dependent on the application and even the geographical deployment of the nodes. Different applications working on the same WSN platform will have different security requirements, implying the necessity of using keys of different length of the applied security algorithms. Usage of smaller key sizes for less sensitive information is an adequate means to minimize the energy consumption. However, many of today's efficient cryptographic implementations do not allow changing the key size of the security algorithm once the system is deployed in the field. That is particularly true for optimized hardware-based cryptographic accelerator platforms as they are realized as ASICS to improve the performance and reduce the energy consumption. Consequently, in spite of the advantages provided by hardware (HW) implementations, only software (SW)-based solutions allow configuration of different security levels in run time. In contrast to pure hardware solutions, including an FPGA that supports partial reconfiguration capabilities into a

sensor node makes feasible the computational advantages of hardware implementations and the flexibility of software. If dynamic and partial reconfiguration is performed remotely, in an autonomous way, the opportunities to improve the security and reliability of the system once deployed increase remarkably.

To demonstrate and evaluate that approach, this paper focuses on the implementation of the scalar point multiplication, which is the fundamental operation of the public-key Elliptic Curve Cryptography (ECC). This cryptosystem is chosen due to the tradeoff between high security and good performance offered by ECC, compared with other state-of-the-art alternatives, like RSA. Furthermore, this operation can be used in digital signature (ECDSA), key establishment (ECDH) and encryption/decryption (ECIES) protocols. The system has been implemented in a custom platform, which incorporates a mixed solution based on an 8052 compliant microcontroller and a Xilinx XC3S200 Spartan 3 FPGA. An additional XC2V2000 Virtex 2 FPGA was attached to the custom platform due to size limitations. This approach is far from being optimized in hardware, but it is suitable for proof of concept.

Beside the above mentioned, partial dynamic reconfiguration techniques are applied to the system. These techniques allow reconfiguring only part of the FPGA without stopping the rest of the circuit. The target is to adapt sensor networks to different security requirements, depending on the application and energy budget. Finally, a software implementation of the ECC algorithm was integrated on a TI MSP430 microcontroller, in order to verify and compare the performance between the software framework and the reconfigurable hardware solution.

The rest of the paper is organized as follows: in Section 2, an analysis of the state of the art for security in sensor networks and background on ECC algorithm is presented. In Section 3, a review on platforms for wireless sensor networks is presented, with special emphasis on those that include configurable HW in different ways. In Section 4 the EC Cryptosystem Architecture is detailed. In Section 5, the experimental setup used for the different proofs is presented. In Section 6, the measurement results obtained are shown and discussed, and finally, in Section 7 some conclusions are drawn after the description of all the work done.

2. State of the Art

In this section, a state-of-the-art review on security in sensor networks as well as a background on the ECC algorithm is provided.

2.1. Security on Sensor Networks. WSN security challenges have been addressed with different approaches in the recent past. Specific schemes and techniques have been developed for WSN scenarios. They are mainly based on secret key cryptography, since public key-based ones [1] have a computation overhead that has been thought to be infeasible sensor nodes. An example of a protocol based on symmetric cryptography is SPINS [2], including TESLA, that provides authenticated streaming broadcast, and SNEP, addressing
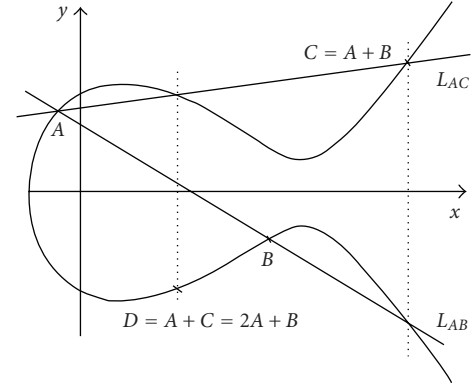

FIGURE 1: EC point addition: the summation of the two points A and B geometrically constructed by determining third intersection point (L_{AB}) of the line through A and B and mirroring this point on the x-axis to achieve $C = A + B$.

data confidentiality, data authentication, and data freshness, based on a subset of the RC5 algorithm. Other examples are TinySEC [3], a link layer security architecture that includes cryptographic primitives like Skipjack and RC5, and MiniSEC [4], a general purpose security protocol for the Telos motes, using a Skipjack block-cipher and OCB encryption.

The main problem when using symmetric security architectures in WSNs is key establishment and distribution. Traditional approaches are based on trusted centralized distribution sites, like Kerberos. These approaches are not appropriate for sensor networks due to the dynamic nature of the networks, the random behavior of the communication links, and the risk of relying on a single point, located in a potentially aggressive environment. The most straightforward solution for this problem is to perform a key predistribution before deployment [5]. A single key is stored in all nodes, thus capturing a single node compromises all other nodes as well, which means weak security. Random key distribution, originally proposed by Eschenauer and Gligor in [6] is a probabilistic key predistribution scheme, based on the storage, in each node, of a random subset from the full pool of keys, reducing memory requirements and consequences of the capture of a node. If the secure connectivity of the whole network is still retained using the initially defined secure paths, a key-exchange mechanism can be employed to deliver a key between any arbitrary pair of nodes. Different extensions of this approach, that increase the security or the efficiency, are summarized in [7], like the q-composite or the random pairwise schemes of Chan et al. [8], and the Du et al. alternative introduced in [9]. More advanced solutions, based on network clustering schemes, are LEAP, a flexible protocol with support for different security level keys [10], SHELL, with high robustness against node capture [11], and Panja hierarchical key-grouping scheme, using the tree-based Diffie-Hellman protocol [12].

All these algorithms and techniques have been specifically designed or adapted for sensor networks, making some simplifications that can compromise its security. In addition,

they are only partially scalable, since predictions about future deployments have to be done before the distribution and its memory and bandwidth consumption is still high. Furthermore, they cannot provide the advantages of public-key algorithms and architectures, like digital signature or session key distribution, well known and successfully proved in traditional networks.

Public key encryption schemes can be used to establish private keys between nodes after deployment, using the less complex private schemes to encrypt and decrypt data, allowing decentralized and completely scalable solutions. For instance, in [13], a Diffie-Hellman exchange protocol has been implemented on WSN, based on a software ECC version implemented for MICAz. Another novel approach is the access control scheme for sensor networks provided in [14].

Public cryptography, for years, has been considered infeasible, because of their cost and speed, to be implemented on sensor networks. However, recent works show successful implementation examples of public-key cryptography in constrained embedded devices [15]. The well-established asymmetric cryptosystem RSA—RSA stands for Rivest, Shamir, and Adleman who first published the algorithm—uses 1024 bit and 2048 bit keys and is 1000 times slower than a symmetric cryptosystem. RSA is based on the factorisation of large prime numbers. Elliptic curve cryptography (ECC), a modern replacement of RSA, achieves the same level of security with a much smaller elliptic curve group. Using a smaller group reduces the requirements for storage and transmission. Almost all the ECC implementations reported for constraint embedded devices are software libraries of different versions of the algorithm, like NanoECC [16], using the ATmega128L and MSP430F1611 processors, or TinyECC [17], or the approaches in [18]. In [19], a very efficient ECC implementation called WM_ECC, which is based on prime field operations, is reported. Solutions differ on the implementation algorithms, the optimizations performed, the functional completeness and platforms. The maximum key size reported in all these articles is 192 bits. The power consumption of these software solutions is still too high for applications on real world commercial solutions.

An alternative is to include hardware accelerators to increase the speed and the efficiency of Public Key cryptography on sensor networks. One of the first results is secFLECK [20], a platform that includes a closed Trusted Platform Module (TPM) from ATMEL, implementing the RSA algorithm. Other example is the smartcard-based solution, proposed in [21]. In [22], a platform with reconfigurable capabilities is used to implement the Rabin's public cryptography scheme, with a working principle similar to RSA, on the FPGA included in the mote. This approach, due to the asymmetric cost of decryption and encryption, is well suited to environments that have much more encryption requirements than decryption.

So, there is no doubt about the benefits of applying asymmetric encryption algorithms with variable key length in WSN technologies. The main obstacles are solved by using reconfigurable HW resources. These show unprecedented speeds (due to the use of HW) and, as it will be shown, better energy usage, and with the flexibility comparable to the one of SW due to the use of reconfigurable logic. This will be the core of the work presented in the following sections.

2.2. Elliptic Curve Cryptosystem. The elliptic curve cryptosystem (ECC) is an asymmetric cryptosystem that uses algebraic operations on elliptic curves (EC) over finite fields. System's security is based on the complexity of finding the discrete logarithm of a random elliptic curve element. Due to the higher difficulty of calculating a discrete logarithm for points of elliptic curves than the factorisation of large prime, much smaller keys than in RSA can be used. The level of security provided by a 1024 bit RSA key can be achieved by a 160 bit EC key.

Arithmetic Basics. ECC is a cryptography scheme that uses points (x, y) on elliptic curves over finite fields. Any EC point meets the function

$$y^2 + xy = x^3 + x^2 + b, \tag{1}$$

where b is a parameter that characterizes the curve. For the points—elements of the finite field—mathematic operations, like addition, multiplication, and so forth, are defined. Figure 1 shows an exemplary EC and illustrates the point addition on such a curve. For cryptographic computations the scalar multiplication kP is the most common operation, while k is an integer and P is a point (x, y) of the elliptic curve. With repeated double and add operations the scalar multiplication can be efficiently calculated, while its reversal is not computable in polynomial time.

For ECC, two different kinds of base fields are suitable, the residue class fields of large prime numbers $(GF(p))$ and residue class fields of extended binary fields $(GF(2^m))$. Both fields are classified as secure, but suitability for implementation differs in hardware and software. Regarding software implementations, the arithmetic in $GF(p)$ is performed with natural integers, which simplifies the implementation in standard microprocessors and makes the use of coprocessing units, like multiplier, easier. However, an energy- and space-efficient hardware implementation is more feasible with binary fields. Major properties are as follows.

(i) Addition and subtraction are replaced by XOR operations.

(ii) Carry bits, which slows down integer arithmetics, can be ignored.

(iii) The binary coefficients can be efficiently represented in hardware.

Consequently, the implementation described in the next sections is based on binary fields.

3. Platforms for Wireless Sensor Networks

There is a huge variety of wireless sensor network nodes at commercial or academic levels. For the purpose of this paper, they can be classified, according to the nature of the processing elements they integrate, into the following categories:

(i) based on a single microprocessor,

(ii) based on FPGAs,

(iii) based on microcontroller plus FPGA,

(iv) based on a System on Programmable Chip,

(v) based on custom configurable HW and a microprocessor.

In this section, several platforms for wireless sensor networks are presented, focusing on those that include configurable HW. Last subsection presents the custom platform which has been used for this work.

3.1. Nodes Based on a Single Microprocessor. Traditionally, wireless sensor networks platforms have been based on low-cost microcontrollers. This approach is based on the idea that nodes do not have to perform highly complex tasks and, moreover, the system should be sleeping most of the time, to save energy.

Good examples of this kind of platforms are TelosB from U. C. Berkeley [23], Imote2 from Intel [24], or the Hitachi one, ZN1 [25]. These nodes have, as main features, low complexity, low-power consumption, and low size.

Since these platforms do not have reconfigurable resources, they are out of the scope of this paper, and so, the focus will center on new solutions with some kind of combinations of reconfigurable devices of different nature and microcontrollers/microprocessors.

3.2. Nodes Based on FPGA. In this section, both pure HW approaches as well as solutions with embedded processors in the FPGA will be taken into account.

At Technical University of Crete, researchers propose a low-cost distributed environment for reconfiguring programmable distributed systems, like sensor networks, called Parrotfish [26]. In their work, they propose an architecture divided in three layers: physical and link layer (with Bluetooth technology), intermediate layer, to control the node, and reconfiguration layer.

The Institute of Microelectronics System of Darmstadt University, has developed a hardware platform based on an FPGA [27]. The FPGA is the heart of the design, and it is used to integrate debugging and system monitoring in the logic, and to emulate the digital part of the final node. Therefore, their main target is prototyping.

Other groups focus on mixed developments based on hardware-software codesign, but everything integrated within the FPGA, like Muralidhar and Rao [28]. In this work the authors used an FPGA from Altera, a Cyclone II, and integrate a soft-processor, a NIOS. Their target is to reuse hardware by dynamic reconfiguration.

Another interesting approach is made by Kahn and Vemuri [29]. In this work, the influence of adding reconfigurable hardware to WSNs nodes is analyzed. Their main contribution is to establish a very accurate model of the battery, what is usually ignored in most of the works related to WSNs, even in those works that consider the power consumption or analyze it. When the battery is about to wear
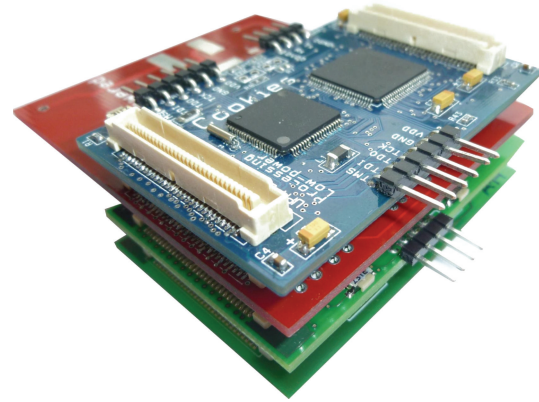

FIGURE 2: HW platform used, with μC, and FPGA.

out, they reconfigure the FPGA with less power consuming tasks, until energy is finished.

3.3. Nodes Based on Microcontroller Plus FPGA. In this context, Wilder et al. present [30] a reconfigurable wireless sensor network (REWISE), showing that FPGAs reach an optimum balance between processing power, energy requirements, and flexibility. Using reconfigurable HW, designs within the nodes can be reprogrammed in the field.

Their solution is based in a 16-bits microcontroller (MSP430) and a CC2420 transceiver, both from Texas Instruments. Their system is an implementation of what they call wireless JTAG, which is independent of the final FPGA that will be used. Each node includes a repository of HW/SW configurations and programs to avoid the high cost of sending these files through the network.

The approach in [31], from Tyndall Institute in Cork, sets out a modular platform based on an Atmel microcontroller (ATMEGA128), with the possibility of adding an FPGA (Spartan 2 XC2S300E from Xilinx) to have more processing power, like Fast Fourier Transforms or neural networks. Anyway, the work is not focused on FPGA reconfiguration, but reprogramming both microcontroller and FPGA before deploying the WSN.

3.4. Node Based on System on Programmable Chip. The third group is that which include System on Programmable Chips (SoPCs), systems with a reconfigurable part, and a processor, everything in the same integrated circuit. This approach is very interesting because there are very attractive commercial solutions from manufacturers like Atmel or Cypress, although the main drawback is the reduced size that can be a bottleneck in applications with high area requirements.

In [32] an implementation based on a SoPC from Atmel (a Field Programmable System Level Integrated Circuit, FPLSIC), composed of an 8-bits AVR processor and 40 kgates of reconfigurable logic, is presented. In this work, one of the factors to highlight is the energy scavenging from the environment, which totally determines the reconfiguration mechanism of the network nodes. In this context, an

algorithm to adapt the node (reconfiguring the HW) is implemented. Finally, they have a solution with the speed of HW and the flexibility of SW. Their conclusion is that runtime reconfigurability in wireless sensor networks is feasible and useful.

A similar approach is adopted by Şuşu et al. in [33], where a video application is implemented with a camera using an FPLSIC, fully powered by energy from the environment.

3.5. Nodes Based on Custom Reconfigurable HW and a Microprocessor.

Some researchers work on solutions where SW and HW coexist from the conceptual formulation of the circuit, so that there is a reconfigurable element attached to the processor, and it can be used as the designer prefers. This approach is, first of all, flexible.

Hinkelmann et al. propose in [34] an architecture based on a custom self-made processor with embedded reconfigurable hardware logic. They state that HW requirements for WSNs can change due to the nature of the applications or changes in the environment. A flexible node can be configured to be used in several applications and could be potentially deployed more efficiently than specific ones.

Their main contribution is to include a Reconfiguration Function Unit (RFU) in a RISC processor. The RFU is integrated directly in the processor pipeline, in parallel with the ALU, and it is accessed through the processor registers.

3.6. The Custom Used Platform.

The approaches presented before represent the majority of possible combinations in the hardware of WSNs nodes.

In this paper, a custom hardware platform for WSNs [35] is used to implement the ECC algorithm, and to study the contribution of reconfigurability to wireless sensor networks in environments where security is a strong requirement.

The node, called *Cookie*, is based on a modular design, and it is composed of four main layers: communications, processing, sensing/actuating, and power supply (see Figure 2).

The processing layer includes a Xilinx Spartan3 XC3S200 and ADuC841 microcontroller from Analog devices, 8052 core compliant. Therefore, this platform belongs to the category described in Section 3.3, with nodes that combine microprocessors and FPGAs.

4. EC Cryptosystem Architecture

For providing an ECC-based asymmetric cryptosystem, an efficient scalar point multiplication is required. The function has to be solved twice for the ECDSA verify operation and once for an ECDSA sign operation. The implementation must handle a tradeoff between size, memory footprint or number of gates, and execution time. This means that the execution time can be reduced by parallelizing operations—in hardware—and using large arrays of precomputed values—in software.

The approach taken in the work presented in this paper uses the Montgomery multiplication scheme as described

Table 1: ECC code size size of code and data section of ECC test program in bytes and percentage of flash usage.

	Full program (B)		Minimized program (B)	
Text size	8646	3,30%	5720	2,18%
Data size	2716	1,04%	2289	0,87%

in [36]. The algorithm is implemented in software and in hardware in a similar way. The implementation in software as well as in hardware is focused on a flexible and a size optimized design.

As illustrated in Figure 3, a scalar multiplication of an EC point can be divided into the field multiplication and the reduction of the partial product. The multiplication of two numbers with a length of m bits results in a number with a length of $(2m-1)$ bits. Because the elements of the finite field must have a maximal length of m bits, the corresponding element of the finite field has to be determined. The corresponding element of the finite field is the remainder from the division by the irreducible polynomial. This operation is performed by the reduction unit.

4.1. Software System.

The software system has a generic implementation of the Montgomery scalar point multiplication. This implementation is working on curves of different binary fields and uses generic functions for the basic operations XOR (polynomial addition and subtraction), polynomial multiplication and squaring as well as multiplicative inversion. Only the implementation of the reduction is curve specific. This flexible design has a very small footprint and makes the integration of new curves quite easy. Only the reduction and the initialization function have to be adapted. In addition to this an external storage of unused reduction units—for example, as loadable library—is feasible and opens the opportunity for smaller designs.

Figure 4 contains the footprint of a program with the ECC scalar point multiplication developed in this work, which supports the binary fields 163, 233, 283, 409, and 571. A minimized program that holds merely the multiplication and the necessary initialization functions needs only 5.7 kB of flash memory and 2.3 kB of RAM. This is only a bit more than 2% of the available flash memory and less than 1% of RAM of the used microcontroller (MSP430F5438, 256 kB Flash, 16 kB RAM).

The full program, used in the measurement setup, needs 8.6 kB of flash memory and 2.7 kB RAM (see Table 1). The larger flash usage is due to the additional function for entering and leaving the low-power mode as well as interrupting handling.

4.2. Hardware Implementation.

The field multiplication is the most complex operation of the EC scalar multiplication. In case of using the classic school multiplication 65000 XOR and AND operations are needed for multiplication of two 256-bit numbers. By using the iterative Karatsuba-multiplication, the number of AND operations can be reduced to 6561 [37]. However, the number of XOR operations remains unchanged. In the hardware design of
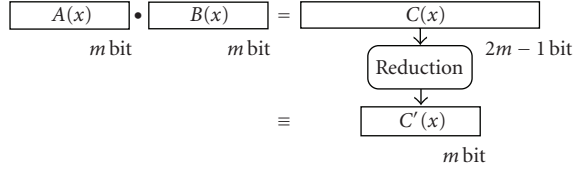
$$A(x) \cdot B(x) = C(x)$$

Figure 3: Multiplication of two m-bit numbers with reduction unit.
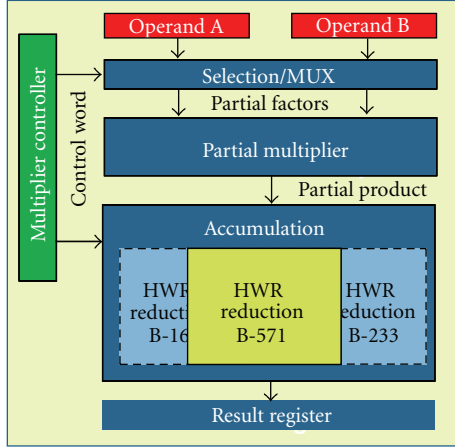
Figure 4: Flexible system design of an ECC coprocessor with replaceable reduction units.

Figure 5: FeuerWhere Node of IHP, a wireless sensor node equipped with three radios, external flash memory and an ultralow power microcontroller.

an EC multiplier presented in the present paper an iterative Karatsuba method is used, in a recursive procedure [38]. This method reduces the number of XOR operations to 42000. The procedure replaces a multiplication with smaller multiplications. The replacement is repeated until a bit length of 8 bits is reached. Finally, the multiplication of the two 8-bit operands will be done by a classic procedure. Evaluations have shown that a further splitting does not improve the procedure [37].

Similar to the software implementation, the hardware field multiplier is generic and supports curve sizes from 163 up to 571 bits. As shown in Figure 4, the hardware unit is managed by a multiplier controller. This controls the input operand selection and the hardware reduction (HWR) unit. The reduction step is curve specific and has to be selected according to the performed operation.

The implementation of such a specific EC coprocessor is feasible and was shown in [38]. The performed size estimations are based on 0.25 μm CMOS design of IHP. The values can be ported to the ECC hardware unit that was used for the measuring in this paper.

In case of using FPGAs for the ECC implementation, the number of used gates has an additional impact. The number of available logic units is limited by the model of the FPGA and can be changed only by switching to another FPGA model with a higher number of logical units (LUT). The need for a larger FPGA will not only influence monetary aspects of the system but also the systems energy consumption. Particularly in case that a model with the required number of LUTs is not available and another FPGA
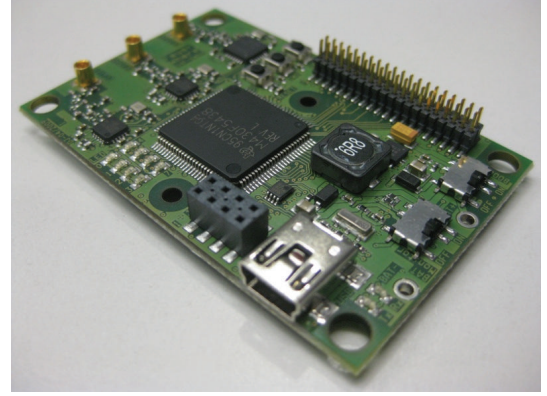
device must be taken. This can be avoided by using the flexible design presented in this paper and reconfigurable FPGA devices, which makes the loading or replacement of partial components like the reduction unit feasible.

5. Experimental Setup

The main objectives of the experimental setup are to compare the performance of both software and hardware implementations of the same ECC algorithm, and to study the feasibility and cost of implementing dynamically reconfigurable ECC algorithms, with different key sizes, on an FPGA. In order to compare in a more realistic scenario, these measurements include the reconfiguration process of the desired encryption algorithm and the execution of the encryption process itself. This scheme also allows sharing the FPGA of the node for different computational tasks. For instance, the ECC algorithm can be implemented during a key establishment phase, to set the private key, and after a reconfiguration process, a less expensive symmetric algorithm can be configured.

In the software experimental setup, ECCs on the binary fields B163, B283, and B571 are used. In a test run, one scalar multiplication of kP is executed, using the base point of the EC as the point P. The scalar value k is defined in such a manner that the first bit is set. Since setting the first bit reductions during multiplication are not possible, an early execution break can be avoided.

The same task has been implemented in hardware, using the efficient architecture described in [35]. Two implementations were chosen with two different reductions: B283 and B571.

In all tests, real power consumption was measured within the sensor nodes or the experimental board, instead of using the information of the corresponding datasheets.

5.1. Software-Based Approach. For the software-based approach, the standard microcontroller MSP430F5438 from Texas Instruments (TI) was used on two different
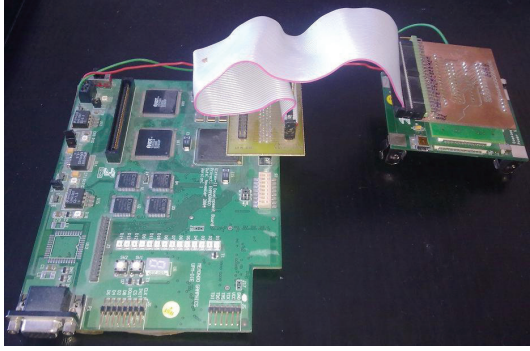
Figure 6: Experimental setup including the expansion FPGA plus the Cookie platform.

boards. The first one is the FeuerWhere Node from IHP (see Figure 5). The node is equipped with the MSP430 microcontroller, three different radio modules, two flash memories, and a USB to serial converter. The second board is the TI experimenter board [35].

The ECC operations were executed in a minimized firmware that initializes all hardware components and runs the tests in an infinite loop. All peripherals on the IHP node are disabled as far as possible. After disabling all peripherals, the node has a power consumption of 2.9 mA in sleep mode. The consumption is caused by nondeactivated debugging functions.

For measurements, start and stop of a single run are signalized via an external General Purpose IO (GPIO) to an external device. The external device—the TI experimenter board is also used here—captures the execution time. The measurement is implemented by using the watchdog timer, which is configured with a resolution of 2 ms.

The power consumption is captured with a digital multimeter "Agilent 34401A". The node is supplied by laboratory power supply that is set to a fixed voltage of 3.3 V. The DC/DC converter on the node is disabled, so that the input voltage for the microcontroller and peripherals is equal to the supply voltage. With this setup it is possible to measure the consumption of all components without any conversion overhead. The TI experimenter board—used for the comparative measurement—is supplied by USB and uses a DC/DC converter to generate the core voltage of 3.3 V. The board is equipped with a jumper to switch off the microcontroller supply. It lies inside the power supply lane to the microcontroller. Using this jumper makes feasible the measurement of the microcontroller power consumption only.

5.2. Reconfigurable Logic Setup. The reconfigurable version of the experimental setup is based on the Cookie platform. The FPGA Spartan 3 XC3S200 included in this platform is not large enough to implement the ECC-571. Consequently, a board with an extra XC2V2000 FPGA was attached to the Cookie (see Figure 6). The FPGA of the sensor node was dedicated only for communication purposes between the ADuC841 microcontroller and the Virtex 2 FPGA, while the ECC core was implemented in this external FPGA.

Additionally, the points of the curve are received using the ZigBee layer of the platform from another external Cookie, in order to set a real scenario. However, the power consumption of the communication layer was ignored for the purpose of this paper. The fact of having the external board constitutes an important overhead regarding the numerical results, but the setup is still a proof of the idea.

Dynamic reconfiguration was used to switch between the two available versions of the hardware core, implementing the elliptic curves B283 and the B571. In this setup, dynamic reconfiguration has been carried out using an external PC, for easiness, although it might be done from the node microcontroller.

6. Measurements

For a comparison of the pure software approach and the custom modular design the power consumption and execution time of both was measured. In this section, measurement results of the software implementation are shown first. The experimental setup explained in Section 5.1 is used and the implementation of ECC point multiplication of Section 4.1. Furthermore the software approach presented in the present work is compared with state-of-the art work to get a ranking of it.

After measuring the software approach, the same operation is run on the Cookie device and power consumption and execution time are captured. In addition to this, setup time of the FPGA was measured to get comparable values for the performance analysis.

6.1. Software Results. The test was executed with four different clock rates for the elliptic curves B163, B283, and B571.

In the tests the power consumption of the IHP node was measured. It was tested in two different execution modes. In the first mode the MSP430 was running in the Low Power Mode 4 (LPM4). In this mode all chip internal peripherals and clocks are disabled and the microcontroller is driven by an external 32 kHz clock. Leaving this mode is possible by generating an interrupt. In the second mode the controller was running at the main clock speed. In this mode four different tests were run to analyze the influence of the main clock speed on the overall power consumption. Tests were executed with the main clock speeds 1 MHz, 4 MHz, 8 MHz, and 16 MHz.

The power consumption in LPM4 is equal to 2.9 mA. In this mode, most of the power will be consumed by the on board peripherals. The microcontroller itself consumes only 41.2 μA in the LPM4. This value was determined on the experimenter board, where it is possible to measure the microcontroller without any peripheral. Nevertheless the consumption is much higher than described in the microcontroller's datasheet. TI specifies a power consumption of 1.69 μA in LPM4. In order to disable the peripherals like radios, flashes, and so forth, some pins have to be held on high level. This causes an additional consumption, which cannot be avoided.

TABLE 2: Measurement results of the scalar multiplication of EC points of curve B163, B283, and B571, executed on the MSP430F5438 on the IHP node.

ECC	MHz	ECC CALC (mA)	Time (ms)	mAs	mWs (3.3 V)
B163	1	3,2	264208	845,47	2790,0
	4	3,8	67272	255,63	843,6
	8	4,6	33748	155,24	512,3
	16	6,1	16904	103,11	340,3
B283	1	3,2	645356	2065,14	6815,0
	4	3,8	164320	624,42	2060,6
	8	4,6	82436	379,21	1251,4
	16	6,1	41292	251,88	831,2
B571	1	3,2	2134901	6831,68	22544,6
	4	3,8	543176	2064,07	6811,4
	8	4,6	272608	1254,00	4138,2
	16	6,1	136598	833,25	2749,7

TABLE 3: Measurement results of ECC multiplication on MSP430 without peripherals.

ECC	MHz	ECC CALC (mA)	Time (ms)	mAs	mWs (3.3 V)
B163	1	0,30	264208	79,26	261,6
	4	0,90	67272	60,54	199,8
	8	1,67	33748	56,36	186,0
	16	3,24	16904	54,77	180,7
B283	1	0,30	645356	193,61	638,9
	4	0,90	164320	147,89	488,0
	8	1,67	82436	137,67	454,3
	16	3,24	41292	133,79	441,5
B571	1	0,30	2134901	640,47	2113,6
	4	0,90	543176	488,86	1613,2
	8	1,67	272608	455,26	1502,3
	16	3,24	136598	442,58	1460,5

As shown in Table 2, the time needed to execute the ECC multiplication ranges from 17 s (B163 at 16 MHz) up to 2135 s (B571 at 1 MHz). Required energy ranges from 0.3 Ws to 22.5 Ws. The minimal execution time and the minimal energy consumption are reached at B163 at 16 MHz. The results show that reducing the clock speed does not improve the energy consumption. In addition, ECC operations on binary fields implemented in plain software and executed with a clock speed less than 8 MHz are not feasible for any application scenario.

In a second test setup, power consumption of the MSP430 without any peripheral was captured. The results are shown in Table 3.

The measurement on the experimenter board substantiates the expectation of the first test. The power consumption is reduced by the idle consumption of the IHP node. Again, the optimum is reached at 16 MHz. For all curves, the

TABLE 4: Performance comparison between state-of-the art works and the proposal in this paper.

	Curve Field	kP Time (s)	ROM size (kB)	RAM size (kB)
Proposal	GF(2^m)	33.7	5.7	2.3
Araz and Qi [39]	GF(2^m)	32.5	20	1.5
TinyECC [17]	GF(2^m)	9.9	12.5	1.3
NanoECC [16]	GF(2^m)	1.1	32.1	2.8
WM-ECC. [19]	GF(p)	0.74	13.8	1.3
NanoECC [16]	GF(p)	0.72	31.3	2.9

multiplication can be executed with a minimum of energy at this clock speed.

In Table 4 the performance of the scalar multiplication is compared with state-of-the art works. All the provided results have been performed under the same conditions. While in [16, 17, 39] 163-bit curves on binary fields are used, [16, 19] demonstrate the performance of ECC implementations for 160-bit curves on prime fields. All the results are based on a pure point multiplication, executed at 8 MHz processor speed on MSP430 microcontroller.

The provided solution in the present work is slower than other more optimized approaches. This is mainly due to the fact that the presented system is intended to be easily ported into a hardware design and require a very small memory footprint. Furthermore, the performance of the software implementation is comparable to [39], which is used for self-certified key establishment methodologies in WSNs. Generally, the performance of the other implementations is very similar to ours, which justifies the comparisons provided in this work.

Beyond that, the NanoECC implementation is optimized for processing speed and the MSP430F1611, which was used in [16], is an ultralow power device. At 8 MHz and a voltage of 3.3 V the microcontroller needs only 1.8 mA. So, the NanoECC and the MSP430 can be seen as the currently least power-consuming (1.872 mAs) software implementation for an EC point multiplication on 163-bit binary fields. But the low power consumption is achieved by using very large memory footprint. With 32.1 kB the algorithm requires 66.9% of the MSP430F1611s and 12.5% of the MSP430F5438s flash memory.

Furthermore as seen in Table 4 the performance of an optimized point multiplication is very similar in binary and prime fields, whereby the WM-ECC implementation extensively uses the hardware multiplier of the microcontroller. As a result of this investigation the least time for software implementation of a point multiplication on an ultra-low power microcontroller is tightly close to one second.

6.2. Hardware Results. The two hardware blocks mentioned before have been implemented in the attached FPGA, obtaining the area occupation results shown in Table 5. As it can be seen in this table, logic resources still remain available to implement other tasks in parallel.

Table 5: Area occupation of the two implemented ECC cores on the Virtex-II FPGA.

	B283	B571	Percentage of the FPGA %
Occupied Slices	7681	14078	54.56
4-Input LUTs	11086	20263	71
Equivalent Gates	98275	180317	48

Table 6: Measurements of the ECC operation time, with different ECC cores and different key sizes for each core.

Hardware block	Size of the key	Time (μs)	Comparative percentage
B283	163	360	10%
	283	750	20.83%
B571	163	370	10.3%
	283	720	20%
	571	3600	100 %

Table 7: Measurements of the dynamic and static power consumption of the DEMO setup, including the Cookie and the attached board.

Hardware block	Dynamic (mA)	Static (mA)	Difference (mA)
B283	206	110	96
B571	231	140	91

In Table 6, execution time for the hardware blocks are shown. The working frequency of all the hardware is 25 MHz. Each HW version can calculate the ECC for the curve equal or less to its size. The B283 is smaller than the B571 version, because it uses smaller registers, busses, and so on. Consequently, regarding the execution time, all the possible combinations of HW block and key sizes have been evaluated.

In Table 7, the measured power consumption of the cores is shown. The power consumption percentage of the ECC283 implementation in comparison to the ECC571 is 89% of the dynamic consumption, and 78% of the Static one. It is remarkable that the difference of energy consumption between 283 and 571 implementations, even though it is not very high, is big enough to justify the reconfiguration of the HW in embedded systems.

In Table 8, a similar analysis has been done, isolating the power consumption of the attached board, that is, of the ECC algorithm itself.

According to Table 8, it is possible to observe that the dynamic configuration is even more justified, in order to minimize the power consumption.

The FPGA included in the attached board cannot be reconfigured using the microprocessor included in the Cookie board. As a result, the functional correctness of the reconfiguration has been carried out using the external PC. However, the timing and power consumption measurements of this approach cannot be extrapolated to the self-reconfigurable autonomous solution. Consequently, the

Table 8: Measurements of the dynamic and static power consumption of the DEMO setup, including only the attached board.

Hardware block	Dynamic (mA)	Static (mA)	Difference (mA)
B283	144	88	56
B571	180	123	57

Table 9: Reconfiguration time estimation of the Spartan3 FPGA.

Hardware block	Slices (CLB columns)	Reconfiguration time (ms)
B283	57	42,24
B571	109	80,77

Table 10: Power consumption estimation of the Reconfiguration of each Cipher Block.

Hardware block	Power consumption (mAs)
B283	253
B571	484

cost of this approach has been estimated, using previously measured results. The reconfiguration time of each column of CLBs of the IHP FPGA, using the microprocessor in the node, is 741 ms. In spite that there is no enough area on the device to implement the cores, the number of columns that they would occupy on the FPGA is calculated, and used to estimate the reconfiguration time, shown in Table 9. This result is meaningful, since a similar Spartan3 family FPGA device with enough size can be integrated in the custom platform, resulting in comparable results. Regarding the power consumption, the current is set constantly, to 88 mA, while the working consumption is 82 mA. Consequently, the reconfiguration cost is 6 mA, independent of the content of the reconfiguration. The final power consumption will depend only on this value, and the reconfigured area. According to this, the power consumption of the process is shown in Table 10.

6.3. Analysis of the Results. Based on results of software and hardware tests shown in the previous subsection, an analysis can be done using a real application, in order to appropriately compare the different approaches.

A real application scenario, like the Elliptic Curve Digital Signature Algorithm (ECDSA), requires one multiplication for signing and two multiplications for verifying. In the best case—B163@16 MHz, the proposed software solution, needs 32 s for signing and 64 s for verifying. The performance optimized implementation of [16] needs more than one or two seconds for these operations, but this performance improvement is achieved with a 5.6 times larger memory footprint. In addition to this, the measurements with different clock speeds have demonstrated that the best energy balance can be reached at the maximum clock speed. The higher power consumption at this clock speed is compensated by shorter execution time. For real application scenarios, both implementations are only suitable with penalties.

In typical application scenarios, sensor nodes are mostly in LPM4, that is, idle mode. In the proposed application scenario, according to the results of the work presented in this paper, speed can be improved in order to use $370\,\mu$s in the signing step and $370\,\mu$s $\times\,2$ in the verification stage, for a similar ECDSA scenario. As it was expected, ciphering time using an HW accelerator is drastically reduced. However, in this paper, it has been shown that using an ECC scheme on a WSN HW platform is feasible, regarding to time execution. Moreover, with these values, the energy consumption is $55.6\,\mu$As, which is much smaller than the presented SW approach (103.11 mAs) and the optimized SW implementation (1.872 mA) presented in [16]. This means $18\cdot10^6$ operations with a 2 Ah standard battery. So, in terms of power consumption, the FPGA-based approach is also feasible.

If this comparison were made for the ECC571, the results would still be better although not so good. In the SW approach, energy consumption is 833.25 mAs and in the HW case 0.8316 mAs, which is a notable improvement, taking into account time execution. Moreover, this curve represents a really high level of security, and still is feasible in the WSN platform.

Considering these results, the FPGA plus μC approach is much more power efficient, taking into account that it is necessary to make the FPGA to go into sleep mode when no processing is required, or even shut it down.

Alternatively, if further processing is required, dynamic configurations is useful, since these HW resources may be reused for other tasks, or adapt one task to different levels of performance, energy consumption, quality, or similar figures. For instance, in the application demonstrator, it has been shown that passing form one ECC implementation into another one has a reconfiguration cost which is worth, since it is performed only once before using the ECC core, in order to have a flexible encryption system that adapts the security level required by the application to the minimum energy usage at all times. The reconfiguration cost is still lower than the software ECC implementations, both in terms of time and power consumption.

7. Conclusions

In this paper, an original implementation of an ECC HW core in wireless sensor networks for a partially and dynamically reconfigurable custom HW platform has been presented.

As a main conclusion of the work presented, it is possible to demonstrate that the HW/SW solution is much more energy efficient than an SW-based one, and even feasible to integrate it in a custom platform for WSNs. It has been demonstrated that a combination of a microcontroller plus FPGA is much faster and energy efficient than a memory footprint and performance optimized software solution.

The measurement results show that a hardware-based ECC implementation executed on an FPGA outperforms software implementations on a low-power microcontroller be at least three orders of magnitude, even when the energy needed to reconfigure the FPGA is taken into account.

Finally, dynamic reconfiguration of the FPGA to carry out other tasks while ciphering is not necessary, but could even further increase the value of the proposed framework for flexible security in WSNs.

For implementing an optimized design for FPGAs, an approach with a generic field multiplier and curve-specific reduction units is proposed as future work. The field multiplier can be used for all curves and is a static part of the FPGA design. The reduction units will be replaceable and will be loaded into the FPGA depending on the application needs. This design takes advantage of the speed, space, and energy optimized, specific design. The different reduction units can be store energy efficient in an external flash memory chip. The overhead in the FPGA for the generic field multiplier and the reload units would be minimal.

Acknowledgment

References

[1] R. R. Brooks, B. Pillai, M. Pirretti, and M. C. Weigle, "Multicast encryption infrastructure for security in sensor networks," *International Journal of Distributed Sensor Networks*, vol. 5, no. 2, pp. 139–157, 2009.

[2] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[3] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *Proceedings of the Second International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 162–175, Baltimore, Md, USA, November 2004.

[4] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "MiniSec: a secure sensor network communication architecture," in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN 2007 '07)*, pp. 479–488, April 2007.

[5] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 53–57, 2004.

[6] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CSS '02)*, V. Atluri, Ed., pp. 41–47, November 2002.

[7] J. Hwang and Y. Kim, "Revisiting random key pre-distribution schemes for wireless sensor networks," in *Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, pp. 43–52, Washington, DC, USA, October 2004.

[8] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of the IEEE Symposium on Security And Privacy*, pp. 197–213, May 2003.

[9] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Transactions on Information and System Security*, vol. 8, no. 2, pp. 228–258, 2005.

[10] S. Zhu, S. Setia, and S. Jajodia, "LEAP+: efficient security mechanisms for large-scale distributed sensor networks,"

ACM Transactions on Sensor Networks, vol. 2, no. 4, pp. 500–528, 2006.

[11] M. F. Younis, K. Ghumman, and M. Eltoweissy, "Location-aware combinatorial key management scheme for clustered sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 8, pp. 865–882, 2006.

[12] B. Panja, S. Madria, and B. Bhargava, "Energy-efficient group key management protocols for hierarchical sensor networks," *International Journal of Distributed Sensor Networks*, vol. 3, no. 2, pp. 201–223, 2007.

[13] S. Khajuria and H. Tange, "Implementation of diffie-Hellman key exchange on wireless sensor using elliptic curve cryptography," in *Proceedings of the 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology (Wireless VITAE '09)*, pp. 772–776, May 2009.

[14] H. Wang, B. Sheng, and Q. Li, "Elliptic curve cryptography based access control in sensor networks," *International Journal of Security and Networks*, vol. 1, no. 3-4, pp. 127–137, 2006.

[15] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and RSA on 8-Bit CPUs," in *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems (CHES '04)*, vol. 3156, pp. 119–132, 2004.

[16] P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier, and R. Dahab, "NanoECC: testing the limits of elliptic curve cryptography in sensor networks," *Proceedings of the 7th international Conference on Information Processing in Sensor Networks (IPSN '08)*, pp. 305–320, 2008.

[17] A. Liu and P. Ning, "TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks," in *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN '08)*, pp. 245–256, 2008.

[18] P. Szczechowiak, A. Kargl, M. Scott, and M. Collier, "On the application of pairing based cryptography to wireless sensor networks," in *Proceedings of the 2nd ACM Conference on Wireless Network Security (WiSec '09)*, pp. 1–12, Zurich, Switzerland, March 2009.

[19] H. Wang and Q. Li, "Efficient implementation of public key cryptosystems on mote sensors," in *Proceedings of the International Conference on Information and Communication Security (ICICS '06)*, pp. 519–528, December 2006.

[20] W. Hu, P. Corke, W. C. Shih, and L. Overs, "SecFleck: a public key technology platform for wireless sensor networks," in *Proceedings of the 6th European Conference on Wireless Sensor Networks*, vol. 5432 of *Lecture Notes in Computer Science*, pp. 296–311, Springer, Cork, Ireland, February 2009.

[21] P. Pecho, J. Nagy, and P. Hanáček, "Power consumption of hardware cryptography platform for wireless sensor," in *Proceedings of the International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT '09)*, pp. 318–323, December 2009.

[22] G. Murphy, A. Keeshan, R. Agarwal, and E. Popovici, "Hardware—software implementation of public-key cryptography for wireless sensor networks," in *IET Irish Signals and Systems Conference*, pp. 463–468, June 2006.

[23] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Proceedings of the 4th IEEE/ACM International Symposium on Information Processing in Sensor Networks (IPSN '05)*, vol. 2005, pp. 364–369, Los Angeles, Calif, USA, April 2005.

[24] R. Adler, M. Flanigan, J. Huang et al., "Intel mote2, an advanced platform for demanding sensor network applications," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys '05)*, p. 298, San Diego, Calif, USA, November 2005.

[25] S. Yamashita, T. Shimura, K. Aiki et al., "A 15×15 mm, 1 μA, reliable sensor-net module: enabling application-specific nodes," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*, vol. 2006, pp. 383–390, Nashville, Tenn, USA, April 2006.

[26] D. Efstathiou, K. Kazakos, and A. Dollas, "Parrotfish: task distribution in a low cost autonomous ad hoc sensor network through dynamic runtime reconfiguration," in *Proceedings of the 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '06)*, pp. 319–320, Napa, Calif, USA, April 2006.

[27] H. Hinkelmann, A. Reinhardt, S. Varyani, and M. Glesner, "A reconfigurable prototyping platform for smart sensor networks," in *Proceedings of the 4th Southern Conference on Programmable Logic (SPL '08)*, pp. 125–130, San Carlos de Bariloche, Argentina, March 2008.

[28] P. Muralidhar and C. B. R. Rao, "Reconfigurable Wireless sensor network node based on NIOS core," in *Proceedings of the 4th International Conference on Wireless Communication and Sensor Networks (WCSN '08)*, pp. 67–72, Allahabad, India, December 2008.

[29] J. Khan and R. Vemuri, "Energy management in battery-powered sensor networks with reconfigurable computing nodes," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '05)*, vol. 2005, pp. 543–546, Tampere, Finland, August 2005.

[30] J. L. Wilder, V. Uzelac, A. Milenković, and E. Jovanov, "Run-time hardware reconfiguration in wireless sensor networks," in *Proceedings of the Annual Southeastern Symposium on System Theory*, pp. 154–158, New Orleans, La, USA, March 2008.

[31] S. J. Bellis, K. Delaney, B. O'Flynn, J. Barton, K. M. Razeeb, and C. O'Mathuna, "Development of field programmable modular wireless sensor network nodes for ambient systems," *Computer Communications*, vol. 28, no. 13, pp. 1531–1544, 2005.

[32] A. Nahapetian, P. Lombardo, A. Acquaviva, L. Benini, and M. Sarrafzadeh, "Dynamic reconfiguration in sensor networks with regenerative energy sources," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '07)*, pp. 1054–1059, Nice, France, April 2007.

[33] A. E. Şuşu, M. Magno, A. Acquaviva, D. Atienza, and G. de Micheli, "Reconfiguration strategies for environmentally powered devices: theoretical analysis and experimental validation," in *Transactions on HiPEAC*, vol. 4050 of *Lecture Notes in Computer Science*, pp. 341–360, Springer, 2007.

[34] H. Hinkelmann, P. Zipf, and M. Glesner, "Design concepts for a dynamically reconfigurable wireless sensor node," in *Proceedings of the 1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS '06)*, pp. 436–441, Istanbul, Turkey, June 2006.

[35] J. Portilla, J. L. Buron, T. Riesgo, and A. De Castro, "A hardware library for sensors/actuators interfaces in sensor networks," in *Proceedings of the 13th IEEE International Conference on Electronics, Circuits, and Systems*, pp. 1244–1247, Nice, France, December 2006.

[36] P. L. Montgomery, "Speeding the pollard and elliptic curve methods of factorization," *Mathematics of Computation*, vol. 48, no. 177, pp. 243–267, 1987.

[37] Z. Dyka and P. Langendoerfer, "Area efficient hardware implementation of elliptic curve cryptography by iteratively applying karatsuba's method," in *Proceedings of the Design, Automation and Test in Europe Conference (DATE '05)*, pp. 70–75, March 2005.

[38] S. Peter and P. Langendörfer, "An efficient polynomial multiplier in GF(2m) and its application to ECC designs," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '07)*, pp. 1253–1258, IEEE Society Press, April 2007.

[39] O. Araz and H. Qi, "Load-balanced key establishment methodologies in wireless sensor networks," *International Journal of Security and Networks*, vol. 1, no. 3-4, pp. 158–166, 2006.