Elliptic Curve Cryptography on the WISP UHF RFID Tag

Christian Pendl, Markus Pelnar, and Michael Hutter

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria
{christian.pendl,m.pelnar}@student.tugraz.at,Michael.Hutter@iaik.tugraz.at

Abstract. The Wireless Identification and Sensing Platform (WISP) can be used to demonstrate and evaluate new RFID applications. In this paper, we present practical results of an implementation of elliptic curve cryptography (ECC) running on the WISP. Our implementation is based on the smallest recommended NIST elliptic curve over prime fields. We meet the low-resource requirements of the platform by various code-size and memory optimizations. Furthermore, we provide a cryptographic framework that allows the realization of different ECC-based protocols on the WISP. We evaluated our implementation results by considering platforms with and without a hardware multiplier. Our best implementation performs a scalar multiplication using the Montgomery powering ladder within 1.6 seconds at a frequency of 6.7 MHz.

Keywords: Public-Key Cryptography, Elliptic Curves, WISP UHF Tag, RFID, Embedded Systems, Privacy.

1 Introduction

Radio-Frequency Identification (RFID) is a wireless technology that allows the communication with passively powered devices. It is a primer for the Internet of Things where many objects are connected to each other and to the Internet to provide a more convenient life to users. With this vision in mind, several security and privacy issues arise that have to be challenged. This paper addresses the implementation of elliptic curve cryptography (ECC) on such platforms in order to overcome these concerns.

Elliptic curve cryptography (ECC) is a public-key technique that has gained much importance due to the high security level while using small key sizes. It is therefore a promising primitive for passive RFID devices to provide various public-key services. These services are for example authentication, confidentiality, non-repudiation, or data integrity. In view of RFID, privacy-preserving authentication is one of the most challenging services. In 2007, S. Vaudenay [31] provided a formal model for RFID protocols and proved that public-key cryptography is required to provide the highest level of feasible privacy in RFID

applications. ECC is a cryptographic primitive that provides a basis for such protocols.

In order to evaluate new RFID protocols and applications, Intel Research Seattle developed a common RFID platform that operates in the UHF frequency range. The Wireless Identification and Sensing Platform (WISP) consists of a tiny low-resource microcontroller that is attached to a dipole antenna. Next to the microcontroller, the tag features several sensors such as temperature, light, and 3D accelerometer which allows a broad range of RFID and sensor node applications. There already exist many publications that use the WISP as a demonstrator platform [24, 25, 32, 27]. Only a few publications presented cryptographic implementations on the WISP such as proposed by H.-J. Chae et al. [2]. They implemented the block cipher RC5 and demonstrated the feasibility of symmetric cryptography on that platform.

In this paper, we present an implementation of elliptic curve cryptography on the WISP. To the authors' knowledge, this is the first publication that demonstrates the feasibility of ECC on that platform. First, we describe several optimizations on the arithmetic level to meet the low-resource requirements of the WISP tag. We apply a hybrid multiplication method that reduces the memory and computational requirements to a minimum. Second, we evaluate the performance of the Montgomery powering ladder based scalar multiplication over the smallest recommended NIST elliptic curve over $\mathbb{F}_{p192}$. Our results show that when running at a frequency of 6.7 MHz WISP tags that do not support a hardware multiplier need 8.3 seconds and only 1.6 seconds when a hardware multiplier is supported. As an outcome, we show that ECC-based RFID protocols can be realized on the WISP and allow the evaluation of new cryptographic implementations for RFID as a proof of concept demonstrator.

The rest of the paper is structured as follows. In Section 2, we give an introduction to elliptic curve cryptography. Section 3 describes the basic features of the WISP tag and the setup which has been used for our experiments. Afterwards, we give implementation details in Section 4. Results of our implementations are given in Section 5. Section 6 concludes the paper and describes future work.

2 Elliptic Curve Cryptography

Each cryptographic principle is based on a mathematical problem that is believed to be "hard". Elliptic curve cryptography relies on the Elliptic Curve Discrete Logarithm Problem (ECDLP) that can only be solved in exponential running time yet. This is a major advantage compared to problems based on the integer factorization or the discrete logarithm problem where subexponential algorithms exist.

An elliptic curve E over a field K is defined with the long Weierstrass equation with the restriction of the discriminant being different from zero which guarantees that the curve is smooth. By using admissible change of variables, a simplified equation can be achieved that is isomorphic to the initial equation

for elliptic curves. Depending on the characteristic of the underlying field K, different cases have to be considered.

In this paper, we deal with elliptic curves over a finite field $\mathbb{F}_q$ of characteristic $\neq 2, 3$ which are defined by the short Weierstrass equation

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

with $a, b \in \mathbb{F}_q$ and the discriminant $\Delta = -16(4a^3 + 27b^2)$. The elliptic curve is defined as a set of points $P = (x, y) \in \mathbb{F}_q$ fulfilling Equation (1). With the point at infinity $\mathcal{O}$, the chord rule for point addition and the tangent rule for point doubling an additive Abelian group is formed. By using this algebraic structure and the two group operations of point addition and point doubling, a scalar multiplication can be performed. A scalar k is multiplied with a point $\boldsymbol{P}$ on the elliptic curve resulting in another point $\boldsymbol{Q} = k \cdot \boldsymbol{P}$. Due to the ECDLP, it is hard to determine k from $\boldsymbol{P}$ and $\boldsymbol{Q}$.

Elliptic-curve points can be represented in different coordinate systems. Affine representation makes use of two coordinates (x, y) to represent a point on the elliptic curve. For the group operations of point addition and point doubling, they require inversions in $\mathbb{F}_q$ which are by far the most expensive field operations. By using projective coordinates, it is possible to avoid such inversions by the costs of an additional coordinate. Homogeneous projective coordinates[1] use three coordinates (X, Y, Z) where $x = \frac{X}{Z}$ and $y = \frac{Y}{Z}$. Jacobian projective coordinates use the relation $x = \frac{X}{Z^2}$ and $y = \frac{Y}{Z^3}$. We refer the reader to e.g. [21, 9] for a more detailed introduction to elliptic curves.

3 WISP UHF RFID Tag

This section gives a brief introduction to the Wireless Identification and Sensing Platform (WISP4.1DL). First, we will describe the hardware of the WISP tag. Second, we will describe the firmware and protocol implementation of the tag. Finally, the reader setup and programming tools that have been used for the implementation are described.

3.1 Hardware

The WISP tags have been developed by Intel Labs Seattle in order to provide a development platform for new RFID and sensing applications. A picture of a WISP4.1DL tag is shown in Figure 1. It is a passively powered RFID tag operating in the UHF frequency range of about 900 MHz featuring an ultra low power general-purpose microcontroller from Texas Instruments [12] (the MSP430F2132). The used microcontroller is a Reduced Instruction Set Computer (RISC) processor providing a 16-bit architecture, 8 KB of flash memory, 512 byte of RAM, and 16 working registers where only 12 can be used for general purpose.

[1] We write affine coordinate in lower case and projective coordinates in upper case.

In particular, the MSP430 family [13] has been especially designed for low-resource applications. It provides various operating modes that can be used to personalize the microcontroller in terms of high speed or low power. Such operating modes range from an active mode (AM) to a low-power mode (LPM4). These different modes basically differ in the number of submodules being disabled to reduce the power consumption of peripherals. In combination with the supported voltage supervisor of the WISP tag, this feature can be used to significantly extend the uptime and reading range of the tag.

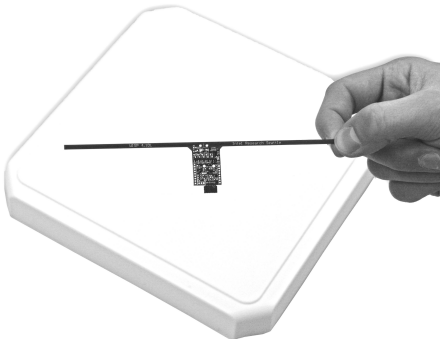

Fig. 1. WISP4.1DL in front of an UHF RFID antenna.

The WISP tag includes several sensors such as a temperature sensor, a light-level detector, and a 3D accelerometer. These sensors allow the realization of a broad range of applications. A. Sample et al. [24] have been the first who reported a WISP-tag application by implementing the symmetric block cipher RC5. N. Saxena and J. Voris [25] extended the use of the accelerometer in order to generate random numbers which are important in the field of cryptography. As another example, D. Yeager et al. [32] extended the RFID antenna in order to serve as a capacitive touch sensor. Because of the combination of both computation and sensing capabilities, WISPs are perfectly suited for human-activity detection as stated by J. R. Smith et al. [27] since they can deliver motion-detection capabilities of active sensor beacons in the same battery-free form factor as RFID tags.

3.2 WISP Firmware

The communication between WISP tags and interrogators (readers) is established over the EPC Class-1 Generation-2 UHF RFID protocol [30]. It has been standardized in ISO/IEC 18000-6C, which defines the physical and logical requirements for a passive backscatter, interrogator-talks-first (ITF) RFID system.

The latest stable release of the firmware—currently the r65 (including the version HW4.1_SW6.0)—provided by Intel Labs Seattle implements main parts of the EPC Class-1 Gen-2 protocol. The firmware allows the configuration of the peripherals such as the temperature sensor or the accelerometer. The sensed data can be transmitted to the reader by either implementing custom commands of the protocol or by using simple EPC read/write commands. Additionally, data encoding can be switched between Miller-2 and Miller-4 modulation with the latter as default setting.

3.3 Reader Setup and Programming Tools

We used the Speedway Revolution R220 [11] UHF reader in our experiments. It supports the EPC Class 1 Generation 2 protocol and provides two high performance monostatic antenna ports. The communication between the reader and a PC is done over a 10/100BASE-T network. Based on this connectivity, the EPCglobal Low Level Reader Protocol (LLRP) v1.0.1 is used as application interface. The transmit power is up to 32.5 dBm using an external power supply.

As a development environment, we used the IAR Embedded Workbench for the MSP430 microcontroller. The flash emulation tool MSP-FET430UIF [29] has been further used to program and debug the WISP tag over an USB interface.

4 Implementation Details

In the following, we give details about the implementation of an ECC framework that runs on the WISP4.1DL RFID tag. Since only low resources are available, several optimizations are necessary to allow the execution of ECC-based protocols on that platform.

One of the most limiting resources of the WISP4.1DL platform is the memory. In fact, the MSP430F2132 is shipped with 8 kB of flash memory where 3.2 kB is needed only for the EPC Class-1 Generation-2 protocol. Thus, implementations are limited to only 4.8 kB. In addition, the MSP430F2132 provides 512 bytes of volatile RAM. The RFID-protocol implementation needs about 200 bytes so that only 312 bytes are available for ECC. Considering these restrictions, it is important to carefully balance between speed and memory consumption. Optimizations such as unrolling of individual operations (such as it is usually the case in finite-field multiplication routines to increase the speed of execution) are therefore not always possible. Since the tag is powered passively, it is furthermore necessary to pay attention to the energy budget. Time-extensive computations need to be separated into parts after which the tag has to test if enough energy is available to continue the operation. If this is not the case, the tag gets into a sleeping mode where the capacitor of the tag can be loaded again to finish the computation.

Another limiting resource in view of ECC is the lack of a dedicated hardware multiplier on the MSP430F2132. In fact, the speed of the multiplication operation largely determines the overall ECC performance. Hardware multipliers can perform a multiplication within only a few clock cycles whereas a few hundred clock cycles are needed if no dedicated multiplier is available. However, many microcontrollers of the MSP430x2xx family feature a hardware multiplier. The work of C. Gouvêa and J. López [6], for example, made use of such a multiplier (multiply-accumulate operation) to speed up the computation on a MSP430 microcontroller. We therefore considered two different implementations for WISP tags. The first implementation provides ECC operations without a multiplier (this is the case for the available WISP4.1DL tag). The second implementation considers a dedicated hardware multiplier and provides ECC operations espe-

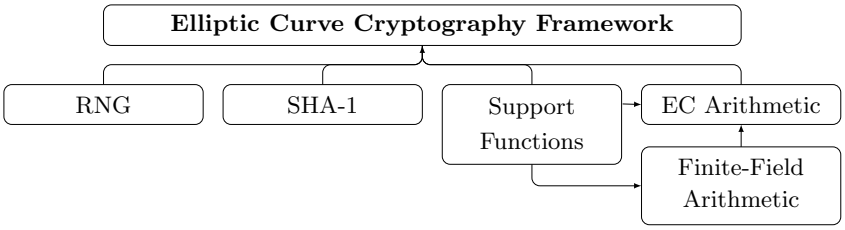

Fig. 2. Overview of the implemented ECC framework.

cially optimized for that scenario (performance results have been simulated in this case).

4.1 Elliptic Curve Cryptography Framework

We implemented a general framework that provides the basic functionalities to implement ECC-based protocols on the WISP tag. Considering security and performance, we decided to base our implementation on a recommended NIST elliptic curve [23]. Due to the limiting resources, we applied the smallest recommended NIST curve over prime fields which is over $\mathbb{F}_{p192}$ where the used prime is a Mersenne-like prime defined as $p \equiv 2^{192} - 2^{64} - 1$.

The entire framework has been implemented in Assembly language to speed-up the computation. As shown in Figure 2, it consists of five main modules: a *random number generator* module, the SHA-1 *hash-function* module, a *support function* module, a *finite-field arithmetic* module, and an *elliptic-curve arithmetic* module.

As a random number generator (RNG), we implemented the algorithm proposed by G. Marsaglia [19, 1]. The algorithm provides a good tradeoff between cryptographic security and computational complexity. It mainly consists of simple shift and xor operations and can therefore efficiently be implemented on the MSP430 microcontroller. For the initialization of the RNG, we used the accelerometer as also proposed by N. Saxena and J. Voris [25] in order to generate a random seed. To guarantee random initialization, the device is locked until stochastic movement of the WISP tag is detected.

We also implemented the SHA-1 hash function in our framework. In fact, hash functions are basic building blocks of cryptographic protocols. They are used, for example, in authentication protocols or digital-signature schemes. Due to memory reasons, we assume that the data which have to be hashed are shorter than the block size of the algorithm, which is 512 bits. This limits the code and memory requirements of our implementation.

The support function module is used to provide basic operations for integer arithmetics such as comparisons ($a \stackrel{?}{=} 0$, $a \stackrel{?}{=} 1$, $a \geq b$), operand copies, or initialization of array elements. These operations are needed mainly for the finite-field arithmetic and the elliptic-curve arithmetic implementation.

The finite-field arithmetic module and the elliptic-curve arithmetic module contain the main operations used to support ECC on the WISP tag. They are described in the following subsections.

4.2 Finite-Field Arithmetic

Both ECC group operations of point addition and doubling are based on underlying finite-field operations. In order to obtain an efficient performance for scalar multiplication, it is important to optimize these operations as much as possible. In fact, finite-field operations are heavily used in loops so that thousands of clock cycles can be saved for scalar multiplication by reducing only one single clock cycle in the underlying finite-field arithmetics.

The MSP430F2132 microcontroller provides a 16-bit architecture. This means that all prime-field operations are based on 16-bit operands, i.e. a 192 bit field element is stored in a 12-word array structure. In particular, we decided to represent each field element in little-endian representation. Thus, indirect autoincrement addressing can be used that is supported by the MSP430 microcontroller. This special addressing mode provides base-address incrementation after the fetch operation without any additional overhead. This allows efficient array processing and avoids additional clock cycles.

Addition and Subtraction. The addition of field elements $a + b = c$ is implemented via a loop that iterates through the twelve words of the operands starting with the least significant word. The operand words are loaded from memory and added using the *ADD* and *ADDC* instruction of the MSP430. In order to speed up the addition operation, we unrolled the loop. In addition to an out-of-place version, we also implemented an in-place version of the prime-field addition where the result overwrites one input operand, i.e. $a \leftarrow a + b$. This has the advantage that special addressing instructions can be used to save execution cycles. In fact, 34 % of the total number of clock cycles can be saved while only increasing the overall code size insignificantly. After the integer addition, the result has to be reduced modulo the prime p. This can be done by simply subtracting the prime if the result is larger than the modulus p. The prime field subtraction has been also implemented by unrolling the instructions. In contrast to modular addition, the modulus p has to be added if the result is smaller than zero.

Multiplication and Squaring. Prime-field multiplication and squaring operations consume most of the running time of a scalar multiplication. They have therefore a significant impact on the overall running time. Consequently, it is crucial to put optimization efforts into these routines. As for modular addition and subtraction, prime-field multiplication and squaring in $\mathbb{F}_p$ consist of a multiplication step (resulting in a double-precision number) and a followed reduction step modulo a prime p.

Two basic multi-precision multiplication algorithms are common: the row-wise standard schoolbook method (also called operand-scanning form) and the column-wise Comba method (also called product-scanning form) [4]. Both algorithms process the words in two loops (an outer loop and an inner loop). In 2004, N. Gura et al. [8] introduced a hybrid method that combines the row-wise and column-wise techniques. The idea behind this method is to make advantage of the two basic multi-precision multiplication algorithms to increase the performance. The Comba method is therefore used in the outer loop of the algorithm and the schoolbook method is used in the inner loop. Still, the number of required registers and needed memory accesses strongly depend on the choice of the algorithm parameter d.

If no hardware multiplier is available on the MSP430, the 16-bit operand-multiplication routine needs four of the twelve registers available. In addition to that, two registers are needed for counter variables as loops have to be used to keep the code size small. Another three registers are necessary to hold the addresses of the operands. With only three remaining registers, it is not possible to implement the hybrid method with parameter $d = 2$ as it would require seven available registers. Thus, for the WISP tag without hardware multiplier, the hybrid method has to be implemented with $d = 1$ which actually corresponds to the standard Comba method. An MSP430 with dedicated hardware multiplier can implement the hybrid method with $d = 2$. This is possible as no additional registers are needed for the 16-bit multiplication. In addition to make the hybrid method with $d = 2$ feasible, we had to use loop unrolling which makes two registers available previously needed for counter variables. As preferable side effect, loop unrolling comes with a significant performance improvement to the cost of substantial code size increase.

For squaring of a field element $c = a^2$, we decided to implement a dedicated squaring operation instead of reusing the multiplication operation. This needs additional code memory but increases the efficiency of scalar multiplication since squaring can be computed faster than multiplication. This is due to the fact that only 78 of the 144 partial products actually have to be computed in case of 192-bit operands and 16-bit wordsize. The remaining partial products can be substituted through cheap shift operations. Additionally, the number of memory accesses and thus required clock cycles can be reduced as only one operand is present. In case of no hardware multiplier the reduction of partial product calculations, which are extremely expensive, clearly outweighs the litte overhead required by the squaring routine. Note that we used the squaring operation only for the implementation of the MSP430 without a hardware multiplier. For the MSP430 with a hardware multiplier, no squaring operation has been implemented. This is due to the fact that the overhead becomes more decisive as the multiplication operation of the dedicated multiplier is much faster than the software multiplication. Thus, a less significant speed improvement is obtained by a squaring operation compared to the MSP430 implementation without a hardware multiplier.

Algorithm 1 Montgomery powering ladder scalar multiplication

Input: $\boldsymbol{P} \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}$, with $k_{n-1} \neq 0$
Output: $\boldsymbol{Q} = k\boldsymbol{P}$

1: $(X_0, Z_0) \leftarrow \boldsymbol{P}$; $(X_1, Z_1) \leftarrow 2\boldsymbol{P}$
2: $X_0 \leftarrow X_0 \times Z_1$; $X_1 \leftarrow X_1 \times Z_0$; $Z \leftarrow Z_0 \times Z_1$;
3: **for** $i = n - 2$ downto 0 **do**
4: $R_2 \leftarrow Z^2$, $R_3 \leftarrow R_2 + R_2$, $R_3 \leftarrow R_3 + R_2$, $R_1 \leftarrow Z \times R_2$, $R_2 \leftarrow 4b \times R_1$,
5: $R_1 \leftarrow {X_{1-k_i}}^2$, $R_5 \leftarrow R_1 + R_3$, $R_4 \leftarrow {R_5}^2$, $R_1 \leftarrow R_1 - R_3$, $R_5 \leftarrow X_{1-k_i} \times R_1$,
6: $R_5 \leftarrow R_5 + R_5$, $R_5 \leftarrow R_5 + R_5$, $R_5 \leftarrow R_5 + R_2$, $R_1 \leftarrow R_1 - R_3$, $R_3 \leftarrow {X_{k_i}}^2$,
7: $R_1 \leftarrow R_1 + R_3$, $X_{k_i} \leftarrow X_{k_i} - X_{1-k_i}$, $X_{1-k_i} \leftarrow X_{1-k_i} + X_{1-k_i}$, $R_3 \leftarrow X_{1-k_i} \times R_2$,
8: $R_4 \leftarrow R_4 - R_3$, $R_3 \leftarrow {X_{k_i}}^2$, $R_1 \leftarrow R_1 - R_3$, $X_{k_i} \leftarrow X_{k_i} + X_{1-k_i}$,
9: $X_{1-k_i} \leftarrow X_{k_i} \times R_1$, $X_{1-k_i} \leftarrow X_{1-k_i} + R_2$, $R_2 \leftarrow Z \times R_3$, $Z \leftarrow x_{\boldsymbol{P}} \times R_2$,
10: $X_{1-k_i} \leftarrow X_{1-k_i} - Z$, $X_{k_i} \leftarrow R_5 \times X_{1-k_i}$, $X_{1-k_i} \leftarrow R_3 \times R_4$, $Z \leftarrow R_2 \times R_5$.
11: *test_power_supply*().
12: **end for**
13: **return** $\boldsymbol{Q} = (X_0, Z)$.

To fit the final result in the underlying field $\mathbb{F}_p$, the 384-bit product or square has to be reduced modulo the prime p. As the used NIST prime p is a generalized-Mersenne prime, the reduced result can be computed by simple additions (fast NIST reduction [9]). The reduction operation can be therefore performed very efficiently.

4.3 Elliptic-Curve Arithmetic

The main operation in ECC implementations is the scalar multiplication $\boldsymbol{Q} = k\boldsymbol{P}$. There exist various algorithms to perform a scalar multiplication. One of the most common methods is the double-and-add (or left-to-right binary method) algorithm. It performs a double operation for every bit of the scalar but performs an addition only if the bit is 1. No addition is performed if the bit is 0. This fact makes such an implementation very efficient but provides information of the secret scalar in physical side channels [15, 18]. By analyzing the running time or the power consumption of the double-and-add implementation, an adversary might identify the value of the scalar which makes an implementation weak in terms of side-channel attacks.

Another scalar-multiplication method resistant to many implementation attacks is the Montgomery powering ladder [22]. Next to the fact that it prevents many attacks due to its regular structure, it allows the computation of group operations without y-coordinates [14]. The entire scalar multiplication can be performed with x-coordinate only operations. This further reduces the memory requirements of our implementation. The Montgomery powering ladder is shown in Algorithm 1 where n denotes the bit size of the prime field, i.e. 192. The point $\boldsymbol{P}$ gets multiplied by the scalar k resulting in the point $\boldsymbol{Q}$. All operations are performed with co-Z coordinate representation [20, 16, 5, 10]. That means that all

points on the elliptic curve share a common coordinate, i.e. Z, which requires to store only three (instead of four) variables throughout the Montgomery ladder.

We applied the formulae of M. Hutter et al. [10] to perform a scalar multiplication. The proposed formulae have been especially designed for low-resource devices and perform all operations *out-of-place* which reduces the memory requirements by one temporary register. We applied Algorithm 5 (cf. [10]), needing ten multiplications, five squarings, and sixteen additions (including subtractions)[2] to perform a (differential) addition and doubling operation. We allocated eight intermediate variables of twelve words each (thus needing 192 bytes of RAM[3] only for the scalar multiplication). In order to keep the memory requirements to a minimum, we decided to reuse four of these intermediate variables also for other operations of the framework.

Due to the high energy costs of a scalar multiplication, we decided to monitor the energy budget during this operation to increase the reading range. Therefore, after each scalar-multiplication iteration a check of the power supply is performed. Thus, the available energy is tested 190 times for one Montgomery-ladder execution. If needed the device is put into sleeping mode to recover energy.

5 Results

In the following, we present experimental results of our ECC framework suitable for WISP. All implementations have been compiled using the IAR Assembler v5.10.4 and the IAR C/C++ Compiler v5.10.6 [Kickstart LMS] that have been configured for low optimization. As stated in Section 4, we provide results of two different ECC implementations. One that has been optimized for WISP tags that feature an MSP430 with no hardware multiplier and one implementation for WISP tags with an MSP430 that supports a hardware multiplier. For a fair comparison of the two implementations, we omitted the EPC Gen-2 protocol implementation for all simulations. On the one hand we used the MSP430F233 microcontroller as a device that features a hardware multiplier, and on the other hand we used the MSP430F2132 that is assembled on the WISP4.1DL tag. To obtain practical results for the WISP4.1DL, the existing firmware has been flattened to allow a running system containing the EPC class-1 Generation-2 protocol as well as the ECC framework.

Most effort in optimizations have been put into the finite-field multiplication and squaring operation. As a multiplication method, we implemented the hybrid method as described in Subsection 4.2. Table 1 shows the amount of required clock cycles for multi-precision multiplication and squaring as a function of the

[2] The multiplication with the curve parameter b (or $4b$) has been realized by a normal multiplication. Multiplication with a has been realized by two additional additions.

[3] Note that the memory requirements can be further reduced to only 168 bytes (cf. Algorithm 4 in [10]). However, this would increase the runtime complexity to twelve multiplications, four squarings, and sixteen additions.

Table 1. Performance of 192-bit multi-precision multiplication and squaring.

System setting	**Multiplication ($a \cdot a$) [Cycles]**	**Squaring (a^2) [Cycles]**
WISP without HWM (d = 1)	25,350	14,361
WISP with HWM (d = 1)	5,046	3,363
WISP with HWM (d = 2)	2,581	-

different system settings. As it was expected, the worst case is the setting without hardware multiplier and the parameter $d = 1$ of the hybrid-multiplication method. The performance can be improved significantly by using a device with hardware multiplier (about 90 % improvement). We have to mention that the implementation of the hybrid-multiplication method with $d = 1$ has not been optimized for usage with hardware multiplier. So there is still a lot of room for optimizations by specially fitting the hybrid-multiplication method to devices that feature a hardware multiplier. As it can be seen in Table 1, the amount of required clock cycles for a multi-precision multiplication can roughly be halved by applying loop unrolling and using $d = 2$. Nevertheless, this optimization is not always practicable as the code size increases significantly as shown in Table 2. The increase in code size is caused by the loop unrolling when applying $d = 2$.

Another improvement can be achieved by introducing squaring (a^2) instead of multiplication ($a \cdot a$). If no hardware multiplier is available, the number of clock cycles can be reduced roughly by the factor 1.7 to the cost of an increased code size. This enormous improvement compared to the implementation of H. Cohen et al. [3] can be explained by the high costs of partial product calculations. For the system settings with hardware multiplier the improvement is less significant as the major improvement through the use of the hardware multiplier itself. So if a computation of the partial products for a multi-precision multiplication is relatively expensive, usage of squaring is recommended because the advantage of clock-cycle reduction outweighs the drawback of an increased code size. Since squaring does not provide a major improvement in running time when compared to the hybrid-multiplication method with $d = 2$, squaring has not been implemented for this system setting.

Table 2. Flash-memory requirements of the multi-precision arithmetic implementation with and without dedicated squarer.

System setting	**Code size without dedicated squarer [Bytes]**	**Code size with dedicated squarer [Bytes]**
WISP without HWM (d=1)	1,076	1,572
WISP with HWM (d=1)	1,020	1,376
WISP with HWM (d=2)	4,236	-

Table 3. Performance of one 192-bit scalar multiplication using the Montgomery ladder running on different WISP-tag settings.

System setting	**Without squaring [Cycles]**	**With squaring [Cycles]**
WISP without HWM (d = 1)	63,257,925	54,630,581
WISP with HWM (d = 1)	17,376,758	15,761,884
WISP with HWM (d = 2)	10,289,883	-

The optimizations described before have been applied on the lowest implementation level. As expected and shown in Table 3, the support of a hardware multiplier provides best performance for the scalar multiplication. If no hardware multiplier is available, a squaring implementation is recommended on the WISP tag as it reduces the amount of clock cycles from 63,257,925 to 54,630,581. Thus, up to 10^7 clock cycles can be saved. Furthermore, it shows that the best performance has been obtained by using a hardware multiplier in combination with the hybrid-multiplication method with parameter $d = 2$. This setting is about six times faster than the fastest method without hardware multiplier.

We simulated the entire 192-bit scalar multiplication using the Montgomery powering ladder. It needs about $5.5 \cdot 10^7$ clock cycles on the WISP4.1DL platform. This corresponds to 8.3 seconds on the WISP running at a clock frequency of 6.7 MHz. Running at this clock frequency—which actually is the highest frequency feasible for succeeding the scalar multiplication with our hardware setup—the maximum distance to the reader is about 2 centimeters. Although the power supply is tested after each Montgomery-ladder round, the scalar multiplication will not finish at distances further than the 2 centimeters as one round

Table 4. Memory requirements of the WISP tag for different framework-level implementations.

Framework level of WISP without HWM	**Code [Bytes]**	**Const [Bytes]**	**Data [Bytes]**
Multi-precision arithmetic	1,572	0	0
Finite-field arithmetic	2,532	0	96
ECC arithmetic	3,944	210	176
Framework level of WISP with HWM (d = 1)	**Code [Bytes]**	**Const [Bytes]**	**Data [Bytes]**
Multi-precision arithmetic	1,376	0	0
Finite-field arithmetic	2,346	0	96
ECC arithmetic	3,758	206	326
Framework level of WISP without HWM (d = 2)	**Code [Bytes]**	**Const [Bytes]**	**Data [Bytes]**
Multi-precision arithmetic	4,236	0	0
Finite-field arithmetic	5,206	0	96
ECC arithmetic	6,618	206	326

Table 5. Memory requirements on the WISP tag for additional framework modules.

Module	Code [Bytes]	Const [Bytes]	Data [Bytes]
SHA-1 hash function	1,012	30	10
Random Number Generation (RNG)	218	0	4
Modified EPC Class 1 Gen 2 protocol	3,260	66	181

exceeds the number of cycles which can be processed with the power available at this distance. If reading ranges larger than the 2 centimeters are necessary, either the clock frequency can be reduced which leads to a longer computation time or an additional capacitor can be assembled on board. Reducing the clock frequency from 6.7 MHz to 6.1 MHz results in a computation time of 9.1 seconds and a maximum distance to the reader of 10 centimeters. Correspondingly, a frequency of 3.98 MHz leads to 13.9 seconds and a frequency of 0.98 MHz leads to 56.2 seconds at a maximum distance of 40 cm. If the MSP430F2132 is replaced with the MSP430F233 and the adaptions for supporting the hardware multiplier are applied, the total number of clock cycles can be reduced by the factor of 5 which results in an estimated computation time of about 1.6 seconds at 6.7 MHz.

Table 4 presents an overview of the required code size for the different system settings. All configurations except the configuration with the hardware multiplier and $d = 2$ support squaring. The code size of the additional framework modules is listed in Table 5.

5.1 Comparison with Related Work

There exist several publications about ECC implementations on the MSP430 family of microcontrollers [6, 7, 26, 17, 28]. Most of the related work make use of the dedicated hardware multiplier that is integrated in many MSP430 architectures. The work of J. Guajardo et al. [7], for example, implemented ECC on the TI MSP430x33x microcontroller. Using an elliptic curve over $\mathbb{F}_{p128}$ they could accomplish a scalar point multiplication in 3.4 seconds at a frequency of 1 MHz. The TI MSP430x33x family features 24 KB or 32 KB of flash memory and 1,024 KB of RAM. In addition, they provide a 16×16-bit hardware multiplier. For a multiplication of two 128-bit operands, only 64 partial products instead of 144 partial products have to be calculated when multiplying two 192 bit operands. Subtraction, addition, and comparison operations also require correspondingly less cycles to complete. As a scalar multiplication method, they applied the double-and-add algorithm.

6 Conclusion

In this paper, we presented practical results of an elliptic curve cryptography (ECC) framework that runs on the Wireless Identification and Sensing Platform (WISP). In order to meet the low-resource requirements of that passively

powered UHF RFID tag, we made several optimizations on arithmetic, algorithmic, and implementation level. We provided results for WISP tags with hardware multiplier such as it is in the case for the WISP4.1DL tag and also for WISP tags which feature a dedicated hardware multiplier, e.g. the MSP430F233 microcontroller. The implementation showed that a scalar multiplication using the Montgomery powering ladder can be performed within 8.3 seconds on the WISP4.1DL tag. The same operation can be performed within 1.6 seconds on the MSP430F233 which is an increase of about 80 %. Furthermore, it showed that a dedicated squaring implementation on the WISP4.1DL improves the performance by about 14 %. Our results show the feasibility of ECC on the WISP while providing a proof of concept demonstrator for future RFID protocols and applications.

As a future work, we plan to evaluate the new generation of WISP tags which provide a Complex Logic Programmable Device (CLPD) on board. This allows to out-source individual operations which can help to reduce the memory requirements and also to improve the performance of ECC implementations. Furthermore, we plan to use the ECC-enabled WISP to evaluate new privacy-preserving authentication protocols. They can be also used to analyze the resistance of such implementations to common attacks.

Acknowledgements.

The work has been supported by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

References

1. R. P. Brent. Note on Marsaglia's Xorshift Random Number Generators. *Journal of Statistical Software*, 11(4):1–5, 8 2004.
2. M.-J. Chae, D. J. Yeager, J. R. Smith, and K. Fu. Maximalist cryptography and computation on the WISP UHF RFID tag. In *Proceedings of the Conference on RFID Security*, 2007.
3. H. Cohen, A. Miyaji, and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates, 1998.
4. P. G. Comba. Exponentiation cryptosystems on the IBM PC. *IBM Syst. J.*, 29:526–538, October 1990.
5. R. Goundar, M. Joye, and A. Miyaji. Co-Z Addition Formulae and Binary Ladders on Elliptic Curves. In S. Mangard and F.-X. Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010, 12th International Workshop Santa Barbara, California, USA, August 1720, 2010 Proceedings*. Springer, 2010.
6. C. Gouvêa and J. Lopez. Software Implementation of Pairing-Based Cryptography on Sensor Networks Using the MSP430 Microcontroller. In B. Roy and N. Sendrier, editors, *Progress in Cryptology - INDOCRYPT 2009*, volume 5922 of *Lecture Notes in Computer Science*, pages 248–262. Springer Berlin / Heidelberg, 2009.

7. J. Guajardo, R. Blümel, U. Krieger, and C. Paar. Efficient Implementation of Elliptic Curve Cryptosystems on the TI MSP 430x33x Family of Microcontrollers. In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*, PKC '01, pages 365–382, London, UK, 2001. Springer-Verlag.
8. N. Gura, A. Patel, A. W, H. Eberle, and S. C. Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. pages 119–132, 2004.
9. D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.
10. M. Hutter, M. Joye, and Y. Sierra. Memory-Constrained Implementations of Elliptic Curve Cryptography in Co-Z Coordinate Representation. In A. Nitaj and D. Pointcheval, editors, *Progress in Cryptology - AFRICACRYPT 2011 Fourth International Conference on Cryptology in Africa, Dakar, Senegal, July 5-7, 2011. Proceedings*, volume 6737 of *Lecture Notes in Computer Science*, pages 170–187, 2011.
11. Impinj. Speedway Revolution - Superior Performance Made Easy, 2010.
12. T. Instruments. MSP430F21x2 Mixed Signal Microcontroller (Rev. G), 2009.
13. T. Instruments. MSP430x2xx Family User's Guide (Rev. F), 2010.
14. M. Joye and S.-M. Yen. The Montgomery Powering Ladder. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 291–302, London, UK, 2003. Springer-Verlag.
15. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, London, UK, 1999. Springer-Verlag.
16. Y. K. Lee and I. Verbauwhede. A Compact Architecture for Montgomery Elliptic Curve Scalar Multiplication Processor. In S. Kim, M. Yung, and H.-W. Lee, editors, *8th International Workshop on Information Security Applications (WISA 2007), Jeju Island, Korea, August 27-29, 2007, Revised Selected Papers*, volume 4867 of *Lecture Notes in Computer Science*, pages 115–127. Springer, 2007.
17. A. Liu and P. Ning. TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. In *International Conference on Information Processing in Sensor Networks - IPSN 2008, April 22-24, 2008, St. Louis, Missouri, USA, Proceedings.*, pages 245–256, St. Louis, MO, April 2008.
18. S. Mangard, M. E. Oswald, and T. Popp. *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer-Verlag, 2007.
19. G. Marsaglia. Xorshift RNGs. *Journal of Statistical Software*, 8(14):1–6, 7 2003.
20. N. Meloni. Fast and Secure Elliptic Curve Scalar Multiplication Over Prime Fields Using Special Addition Chains. Cryptology ePrint Archive, Report 2006/216, 2006.
21. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
22. P. L. Montgomery. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, 48(177):pp. 243–264, 1987.
23. National Institute of Standards and Technology (NIST). FIPS-186-3: Digital Signature Standard (DSS), 2009. Available online at `http://www.itl.nist.gov/fipspubs/`.
24. A. Sample, D. Yeager, and J. Smith. WISP: A Passively Powered UHF RFID Tag with Sensing and Computation. In *RFID Handbook: Applications, Technology, Security, and Privacy*, march 2008.
25. N. Saxena and J. Voris. Accelerometer Based Random Number Generation on RFID Tags. 1st Workshop on Wirelessly Powered Sensor Networks and Computational RFID (WISP Summit), 2009.

26. M. Scott and P. Szczechowiak. Optimizing Multiprecision Multiplication for Public Key Cryptography. Cryptology ePrint Archive (`http://eprint.iacr.org/`), Report 2007/299, 2007.
27. J. R. Smith, K. P. Fishkin, B. Jiang, A. Mamishev, M. Philipose, A. D. Rea, S. Roy, and K. Sundara-Rajan. RFID-based techniques for human-activity detection. *Commun. ACM*, 48:39–44, September 2005.
28. P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier, and R. Dahab. NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks. In R. Verdone, editor, *Wireless Sensor Networks 5th European Conference, EWSN 2008, Bologna, Italy, January 30-February 1, 2008. Proceedings.*, volume 4913 of *Lecture Notes in Computer Science*, pages 305–320. Springer, 2008.
29. Texas Instruments. MSP-FET430UIF, May 2010.
30. The global language of business. EPCTM Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz 960 MHz Version 1.2.0, October 2008.
31. S. Vaudenay. On privacy models for RFID. In *Proceedings of the Advances in Crypotology 13th international conference on Theory and application of cryptology and information security*, ASIACRYPT'07, pages 68–87, Berlin, Heidelberg, 2007. Springer-Verlag.
32. D. Yeager, J. Holleman, R. Prasad, J. Smith, and B. Otis. NeuralWISP: A Wirelessly Powered Neural Interface With 1-m Range. *Biomedical Circuits and Systems, IEEE Transactions on*, 3(6):379 –387, dec. 2009.