Low-Cost Elliptic Curve Cryptography for Wireless Sensor Networks

Lejla Batina, Nele Mentens, Kazuo Sakiyama,
Bart Preneel, and Ingrid Verbauwhede

Katholieke Universiteit Leuven, ESAT/COSIC,
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
{lbatina,nmentens,ksakiyam}@esat.kuleuven.be

Abstract. This work describes a low-cost Public-Key Cryptography (PKC) based solution for security services such as key-distribution and authentication as required for wireless sensor networks. We propose a custom hardware assisted approach to implement Elliptic Curve Cryptography (ECC) in order to obtain stronger cryptography as well as to minimize the power. Our compact and low-power ECC processor contains a Modular Arithmetic Logic Unit (MALU) for ECC field arithmetic. The best solution features 6718 gates for the MALU and control unit (data memory not included) in 0.13 μm CMOS technology over the field $\mathbb{F}_{2^{131}}$, which provides a reasonable level of security for the time being. In this case the consumed power is less than 30 μW when operating frequency is 500 kHz.

Keywords: sensor networks, pervasive computing, Elliptic Curve Cryptography, authentication, key-distribution, hardware implementation.

1 Introduction

The field of embedded security is in constant evolvement and new applications are constantly emerging. Extreme examples are sensor nodes and RFID tags as they put new requirements on implementations of Public-Key protocols with a very low budget for the number of gates, power, bandwidth *etc.* Especially the security in wireless sensor networks is of crucial importance as a large number of nodes is exposed in sometimes hostile environments and if only one node is captured by the attacker, the impact to the complete network can be devastating. Therefore, various cryptographic services are required for these applications and common use of symmetric-key algorithms such as AES and MACs are not just imposing problems such as key protection and management but can be at the same time even more expensive. Although for example, authentication can be obtained by means of symmetric-key cryptography, it is evident that PKC substantially simplifies security protocols. In addition, the use of PKC reduces power due to less protocol overhead [2].

To the best of our knowledge very few papers discuss the possibility for PKC in these applications although the benefits of PKC are evident especially for

L. Buttyan, V. Gligor, and D. Westhoff (Eds.): ESAS 2006, LNCS 4357, pp. 6–17, 2006.

key distribution between the nodes and various authentication protocols. For example, the authentication of the base station is easily performed assuming the public key of the base station can be stored in each node [3]. If only resistance against passive attacks is needed, the algorithm of Schnorr [9] can be used for this purpose as it is known that this scheme is secure against passive attacks under the discrete logarithm assumption. The main cost of this algorithm for the case of ECC is just one point multiplication.

In this paper we investigate the possibility for PK services for pervasive computing. We show that ECC processors can be designed in such a way to qualify for lightweight applications suitable for wireless sensor networks. Here, the term lightweight assumes low die size and low power consumption. Therefore, we propose a hardware processor supporting ECC that features very low footprint and low-power. We investigate ECC over binary fields $\mathbb{F}_{2^p}$ where p is a prime as proposed in standards [4].

The paper is organized as follows. Section 2 lists some related work. In Sect. 3 we give some background information on Elliptic Curve Cryptography and supporting arithmetic. In Sect. 4 we elaborate on a suitable selection of parameters and algorithms and we outline our architecture and describe our hardware implementation. Our results are discussed in Sect. 5. Section 6 concludes the paper.

2 Related Work

Two emerging examples of PKC applications dealing with extremely constrained environments are sensor networks and radio frequency identification tags (RFIDs). They put new requirements on implementations of PK algorithms with very tight constraints in number of gates, power, bandwidth *etc.* Therefore, as related previous work we mention implementations of Public-Key cryptosystems for these applications.

Wireless distributed sensor networks are expected to be used in a broad range of applications, varying from military to meteorological applications [3]. As the current generation is powered by batteries, ultra-low power circuitry is a must for these applications. On the other hand, there is a clear need for PKC in this context, especially for services such as key-exchange protocols that are typically provided by means of PKC.

RFID tags are passive devices consisting of a microchip connected with an antenna. Typically, they have no battery, but they obtain power from the electromagnetic field produced by the RFID reader. Today they are mainly used for identification of products but recent applications include also counterfeiting [10]. The application areas for RFIDs vary from supply chain management, inventory management, preventing banknotes counterfeiting to vehicles tracking, security of newborn babies *etc.* In short, RFID tags are meant to be a ubiquitous replacement for bar codes with some added functionality.

The work of Gaubatz *et al.* [3] discusses the necessity and the feasibility of PKC protocols in sensor networks. In [3], the authors investigated

implementations of two algorithms for this purpose *i.e.* Rabin's scheme and NTRUEncrypt. The conclusion is that NTRUEncrypt features a suitable low-power and small footprint solution with a total complexity of 3000 gates and power consumption of less than 20 μW at 500 kHz. On the other hand, they showed that Rabin's scheme is not a feasible solution. In [2] the authors have compared the previous two algorithm implementations with an ECC solution for wireless sensor networks. The architecture of the ECC processor occupied an area of 18 720 gates and consumed less than 400 μW of power at 500 kHz. The field used was a prime field of order $\approx 2^{100}$.

Some more efforts for PKC processors for RFID tags include the results of Wolkerstorfer [11] and Kumar and Paar [5]. Wolkerstorfer [11] showed that ECC based PKC is feasible on RFID-tags by implementing the ECDSA on a small IC. The chip has an area complexity of around 23 000 gates and it features a latency of 6.67 ms for one point multiplication at 68.5 MHz. However, it can be used for both types of fields *e.g.* $\mathbb{F}_{2^{191}}$ and $\mathbb{F}_{p_{192}}$. The results of Kumar and Paar [5] include an area complexity of almost 12 kgates and a latency of 18 ms for one point multiplication over $\mathbb{F}_{2^{131}}$ at 13.56 MHz. The operating frequency is in both cases too high for those applications and therefore the results cannot be properly evaluated. Namely, with such a high frequency the power consumed becomes too large, which has the most crucial impact on the feasibility of the implementations. We compare the previous implementations with our results in Section 5 in more detail.

3 Elliptic Curve Cryptography

ECC relies on a group structure induced on an elliptic curve. A set of points on an elliptic curve together with the point at infinity, denoted ∞, and with point addition as binary operation has the structure of an abelian group. Here we consider finite fields of characteristic two. A non-supersingular elliptic curve E over $\mathbb{F}_{2^n}$ is defined as the set of solutions $(x, y) \in \mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$ to the equation: $y^2 + xy = x^3 + ax^2 + b$ where $a, b \in \mathbb{F}_{2^n}, b \neq 0$, together with ∞.

The main operation in any ECC-based primitive such as key-exchange or encryption is the scalar multiplication which can be viewed as the top level operation. The point scalar multiplication is achieved by repeated point addition and doubling. All algorithms for modular exponentiation can also be applied for point multiplication.

At the next (lower) level are the point group operations *i.e.* addition and doubling. The point addition in affine coordinates is performed according to the following formulae. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points on an elliptic curve E. Assume $P_1, P_2 \neq \infty$ and $P_1 \neq -P_2$. The sum $P_3 = (x_3, y_3) = P_1 + P_2$ is computed as follows [1]:
If $P_1 \neq P_2$,

$$\begin{aligned} \lambda &= (y_2 + y_1) \cdot (x_2 + x_1)^{-1} \\ x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1 \,. \end{aligned}$$

If $P_1 = P_2$,

$$\begin{aligned} \lambda &= y_1/x_1 + x_1 \\ x_3 &= \lambda^2 + \lambda + a \\ y_3 &= (x_1 + x_3)\lambda + x_3 + y_1 \,. \end{aligned}$$

There are many types of coordinates in which an elliptic curve may be represented. In the equations above affine coordinates are used, but so-called projective coordinates have some implementation advantages. The main conclusion is that point addition can be done in projective coordinates using only field multiplications, with no inversions required. More precisely, only one inversion needs to be performed at the end of a point multiplication operation.

The lowest level consists of finite field operations such as addition, subtraction, multiplication and inversion required to perform the group operations. More details on ECC and its mathematical background can be found in [1].

4 Elliptic Curve Processor (ECP) for Pervasive Computing

4.1 Algorithms Selection and Parameters

For the point multiplication we chose the method of Montgomery (Algorithm 1) [8] that maintains the relationship $P_2 - P_1$ as invariant. It uses a representation where computations are performed on the x-coordinate only in affine coordinates (or on the X and Z coordinates in projective representation). That fact allows us to save registers which is one of the main criteria for obtaining a compact solution.

Algorithm 1. Algorithm for point multiplication

Require: an integer $k > 0$ and a point P
Ensure: $x(kP)$
  $k \leftarrow k_{l-1}, ..., k_1, k_0$
  $P_1 \leftarrow P, \quad P_2 \leftarrow 2P.$
  for i from $l-2$ downto 0 **do**
    If $k_i = 1$ then
    $x(P_1) \leftarrow x(P_1 + P_2)$, $x(P_2) \leftarrow x(2P_2)$
    Else
    $x(P_2) \leftarrow x(P_2 + P_1)$, $x(P_1) \leftarrow x(2P_1)$
  end for
  Return $x(P_1)$

We chose as starting point for our optimizations the formulas of Lopez and Dahab [7]. The original formulas in [7] require 2 or 3 intermediate registers if the point operations are performed sequentially or in parallel respectively. In the case of sequential processing it is enough to use two intermediate variables but

in our case we eliminate one more intermediate register, which added a few more steps to the original algorithms. The results of our optimizations are shown in Algorithm 2.

Algorithm 2 requires only one intermediate variable T, which results in 5 registers in total. The required registers are for the storage of the following variables: X_1, X_2, Z_1, Z_2 and T. Also, the algorithm shows the operations and registers required if the key-bit $k_i = 0$. Another case is completely symmetric and it can be performed accordingly. More precisely, if the addition operation is viewed as a function $f(X_2, Z_2, X_1, Z_1) = (X_2, Z_2)$ for $k_i = 0$ due to the symmetry for the case $k_i = 1$ we get $f(X_1, Z_1, X_2, Z_2) = (X_1, Z_1)$ and the correct result is always stored in the first two input variables. This is possible due to the property of scalar multiplication based on Algorithm 1.

Algorithm 2. EC point operations that minimize the number of registers

Require: X_i, Z_i, for $i = 1, 2$, $x_4 = x(P_2 - P_1)$
Ensure: $X(P_1 + P_2) = X_2$, $Z(P_1 + P_2) = Z_2$
1: $X_2 \leftarrow X_2 \cdot Z_1$
2: $Z_2 \leftarrow X_1 \cdot Z_2$
3: $T \leftarrow X_2 \cdot Z_2$
4: $Z_2 \leftarrow Z_2 + X_2$
5: $Z_2 \leftarrow {Z_2}^2$
6: $X_2 \leftarrow x_4 \cdot Z_1$
7: $X_2 \leftarrow X_2 + T$

Require: $b \in \mathbb{F}_{2^n}$, X_1, Z_1
Ensure: $X(2P_1) = X_1$, $Z(2P_1) = Z_1$,
1: $X_1 \leftarrow X_1^2$
2: $Z_1 \leftarrow Z_1^2$
3: $T \leftarrow Z_1^2$
4: $Z_1 \leftarrow X_1 \cdot Z_1$
5: $T \leftarrow T^2$
6: $T \leftarrow b \cdot T$
7: $X_1 \leftarrow X_1^2$
8: $X_1 \leftarrow X_1 + T$

4.2 Binary Fields Arithmetic

From the formulae for point operations as given in Algorithm 2 it is evident that we need to implement only multiplications and additions. Squaring is considered as a special case of multiplication in order to minimize the area and inversion is avoided by use of projective coordinates. We assume that conversion to affine coordinates can be computed at the base station's side. Note also that, if necessary, the one inversion that is required can be calculated by use of multiplications. In this way the area remains almost intact and some small control logic has to be added.

4.3 Global Architecture

Our Elliptic Curve Processor (ECP) is shown in Fig. 1. The operational blocks are as follows: a Control Unit (CU), an Arithmetic Unit (ALU), and Memory (RAM and ROM). In ROM the ECC parameters and the constants x_4 and b are stored. On the other hand, RAM contains all input and output variables and it therefore communicates with both, the ROM and the ALU.

The Control Unit controls the scalar multiplication and the point operations. In addition, the controller commands the ALU which performs field multiplication, addition and squaring. When the START signal is set, the bits of $k = \sum_{i=0}^{n_k-1} k_i 2^i$, $k_i = \{0, 1\}$, $n_k = \lceil \log_2 k \rceil$, are evaluated from MSB to LSB resulting in the assignment of new values for P_1 and P_2, dependent on the key-bit k_i. When all bits have been evaluated, an internal counter gives an END signal. The result of the last P1 calculation is written to the output register and the VALID output is set. The CU consists of a number of simple state machines and a counter and its area cost is small. The processor memory consists of the equivalent to five n-bit ($n = p$) registers.

As our ALU deals with modular arithmetic in a binary field we refer to it from now on as the Modular Arithmetic Logic Unit (MALU) for which give more details in the following section.

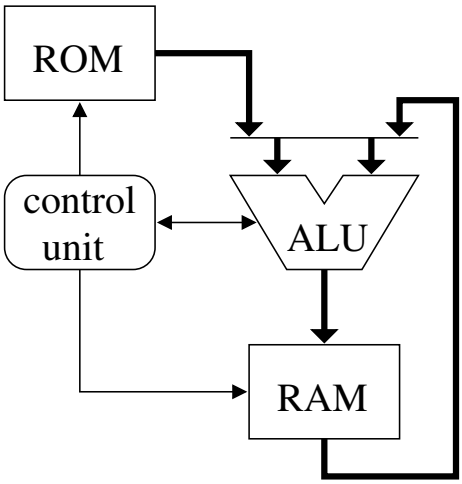

Fig. 1. ECP architecture

4.4 Modular Arithmetic Logic Unit (MALU)

In this section the architecture for the MALU is briefly explained. The datapath of the MALU is an MSB-first bit-serial $\mathbb{F}_{2^n}$ multiplier with digit size d as illustrated in Figure 2. This arithmetic unit computes $A(x)B(x)$ mod $P(x)$ where $A(x) = \sum a_i x^i$, $B(x) = \sum b_i x^i$ and $P(x) = \sum p_i x^i$. The proposed MALU computes $A(x)B(x)$ mod $P(x)$ by following the steps: The MALU_n sums up three types of inputs which are $a_i B(x), m_i P(x)$ and $T(x)$, and then outputs the intermediate result, $T_{next}(x)$ by computing $T_{next}(x) = (T(x) + a_i B(x) + m_i P(x))x$ where $m_i = t_n$. By providing T_{next} as the next input T and repeating the same computation for n times, one can obtain the multiplication result.

Modular addition, $A(x) + C(x)$ mod $P(x)$ can be also supported on the same hardware logic by setting $C(x)$ to the register for $T(x)$ instead of resetting

register $T(x)$ when initializing the MALU. This operation requires additional multiplexors and XORs. However the cost of this solution is much cheaper compared to the case of having a separate modular adder. This type of hardware sharing is very important for such low-cost applications.

The proposed datapath is scalable in the digit size d which can be determined arbitrary by exploring the best combination of performance and cost.

In Fig. 2 the architecture of our MALU is shown for finite fields operations in $\mathbb{F}_{2^{163}}$. To perform a finite field multiplication, the *cmd* value should be set to 1 and the operands should be loaded into registers A and B. The value stored in A is evaluated digit per digit from MSB to LSB. We denote the digit size by d. The result of the multiplication will be provided in register T after $\lceil \frac{163}{d} \rceil$ clock cycles. A finite field addition is performed by giving *cmd* the value 0, resetting register A and loading the operands into registers B and T. The value that is loaded into T is denoted by C. After one clock cycle, the result of the addition is provided in register T. The *cmd* value makes sure that only the last cell is used for this addition.

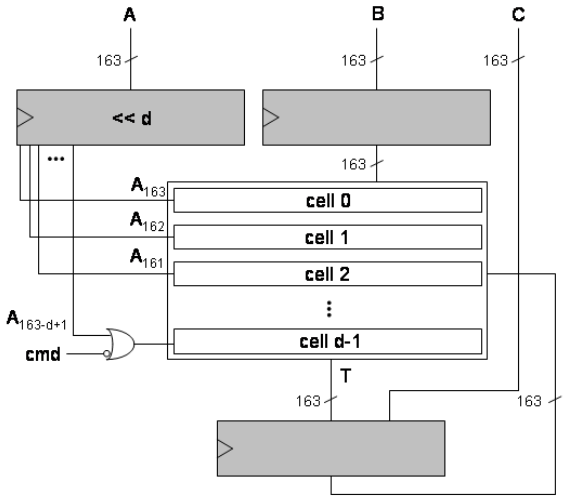

Fig. 2. Architecture of the MALU

The cells inside the MALU all have the same structure, which is depicted in Fig. 3. A cell consists of a full-length array of AND-gates, a full-length array of XOR-gates and a smaller array of XOR-gates. The position of the XOR-gates in the latter array depends on the irreducible polynomial. In this case, the polynomial $P(x) = x^{163} + x^7 + x^6 + x^3 + 1$ is used. The *cmd* value determines whether the reduction needs to be done or not. In case of a finite field multiplication, the reduction is needed. For finite field addition, the reduction will not be performed.

The output value T_{out} is either given (in a shifted way) to the next cell or to the output register T in Fig. 2. The input value T_{in} is either coming from the previous cell or from the output register T.

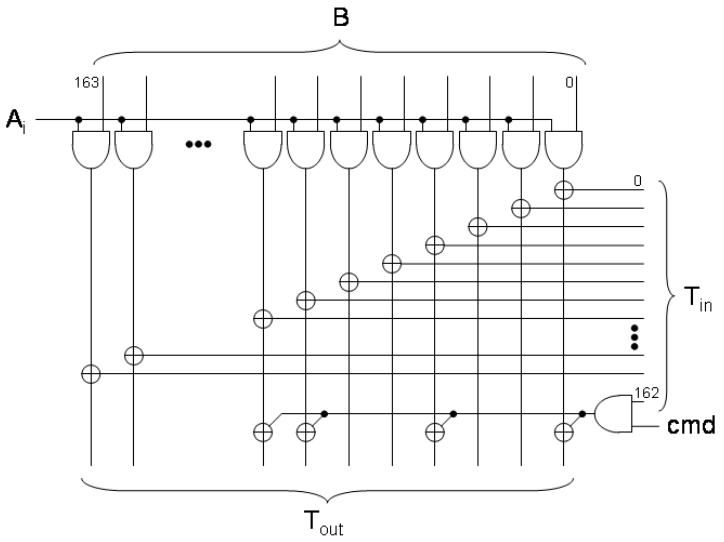

Fig. 3. Logic inside one cell of the MALU

The strong part of this architecture is that it uses the same cell(s) for finite field multiplication and addition without a big overhead in multiplexors. This is achieved by using T as an output register as well as an input register. The flip-flops in T are provided with a load input, which results in a smaller area overhead compared to a solution that would use a full-length array of multiplexors.

5 Results and Discussion

Now we give the results for area complexity and the latency in the case of ECC point multiplication. The designs were synthesized by Synopsys Design Vision using a 0.13 μm CMOS library. We used binary fields from bit-size 131 to 163 as recommended by NIST. ECC with key sizes of around 160 bits is usually compared with RSA for 1024 bits although those are only rough estimates. Namely, according to the work of Lenstra and Verheul 163 bit long key sizes for ECC correspond to RSA keys that are much longer than 1024 bits [6]. More precisely, one could achieve that level of security with around 130 bits long ECC keys. Therefore, we can assume that ECC over $\mathbb{F}_{2^{131}}$ provides a good level of security for these applications.

The results of the area complexity for various architectures with respect to the choice of fields and the size of d for the MALU are given in Table 1. The

Table 1. The area complexity of MALU in gates of the ECC processor for various fields and digit sizes

Field size	d=1	d=2	d=3	d=4
131	4446	4917	5376	5837
139	4716	5214	5712	6189
151	5117	5652	6187	6700
163	5525	6105	6685	7243

Table 2. The complete area complexity in gates of the ECC processor for various fields and digit sizes

Field size	d=1	d=2	d=3	d=4
131	6718	7191	7645	8104
139	7077	7635	8132	8607
151	7673	8205	8738	9252
163	8214	8791	9368	9926

Table 3. The complete area complexity in μm^2 of the ECC processor for various fields and digit sizes

Field size	d=1	d=2	d=3	d=4
131	34936.7	37395.6	39754.4	42139
139	36802.9	39702.5	42287.6	44755.2
151	39901.2	42666	45439.5	48109.2
163	42714.4	45714.2	48715.8	51617.1

results for the complete architecture in gates and in μm^2 are given in Table 2 and Table 3 respectively.

The graphical representations of our results for area are shown in Fig. 4 and Fig. 5. We can observe that the upper bound for the area of the MALU is slightly more than 7 kgates. On the other hand the complete area, so MALU and the CU together is less than 10 kgates.

The graphical representations of our results for area in μm^2 and for the total power consumed are shown in Fig. 6 and Fig. 7. The power estimates were made assuming the operating frequency of 500 kHz. With this frequency the power stays between 20 and 30 μW which is assumed to be acceptable for sensor networks applications.

Next we give the numbers for the performance. For the point multiplication we used Algorithm 1 and for point operations Algorithm 2. We calculate the total number of cycles for each field operation by use of the following formulae for field operations. The total number of cycles for one field multiplication is $\lceil \frac{n}{d} \rceil + 3$ where n and d are the bit size of an arbitrary element from the field in which we are working and the bit size respectively. On the other hand, one field addition takes 4 cycles. The number of cycles required for one point multiplication in the

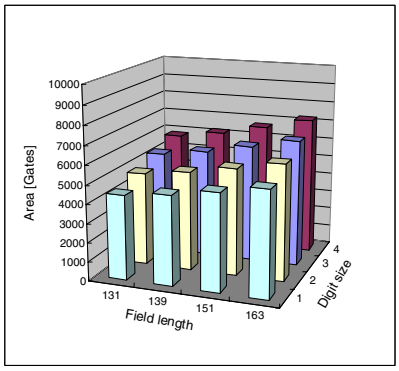

Fig. 4. Results for area complexity of the ECC-dedicated MALU for various fields and digit sizes

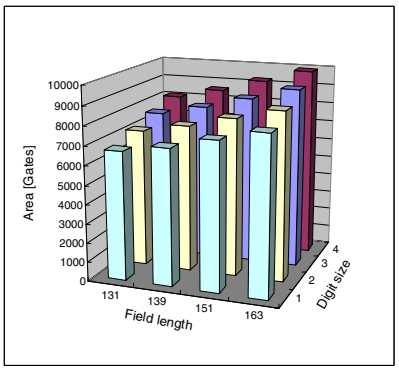

Fig. 5. Results for complete area complexity of ECC processor for various fields and digit sizes

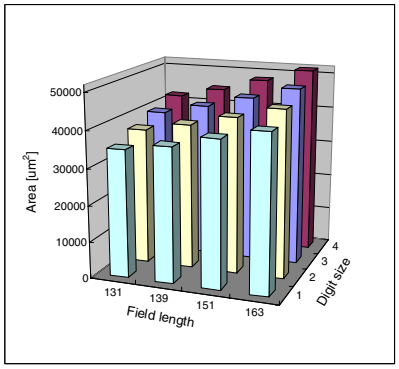

Fig. 6. Results for area complexity in μm^2 of the ECC-processor for various fields and digit sizes

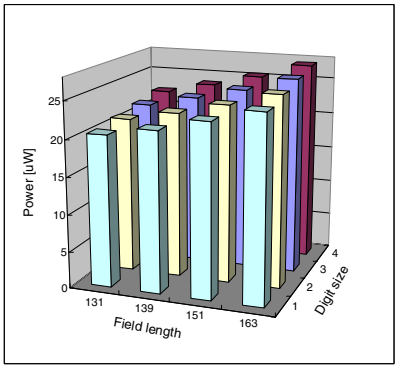

Fig. 7. Results for the power consumed by the ECC processor for various fields and digit sizes

case of field $\mathbb{F}_{2^p}$, where p is a prime is: $(n_k - 1)[13(\lceil \frac{(n_k-1)}{d} \rceil + 3) + 12]$. Here, n_k denotes the number of bits of the scalar k *e.g.* the secret key.

The results for the total number of cycles of one point multiplication for fields $\mathbb{F}_{2^{131}}$ and $\mathbb{F}_{2^{163}}$ are given in Table 4. To calculate the time for one point multiplication we need an operating frequency. However, the frequency that can be used is strictly influenced by the total power. We assumed an operating frequency of 500 kHz as suggested in [3] in order to estimate the actual timing. We get 115 ms for the best case of ECC over $\mathbb{F}_{2^{131}}$ $(d = 4)$ and 190 ms for the best case of ECC over $\mathbb{F}_{2^{163}}$ $(d = 4)$. Our results are compared with other related work in Table 5.

Table 4. The number of cycles required for one point multiplication for ECC over fields $\mathbb{F}_{2^{131}}$ and $\mathbb{F}_{2^{163}}$

Field size	d=1	d=2	d=3	d=4
131	210 600	109 200	74 880	57 720
163	353 710	182071	124 858	95 159

Table 5. Comparison with other related work

Ref.	Fin. field	Area [*gates*]	Techn. [μm]	Op. freq. [kHz]	Perf. [ms]	Power [μW]
[5]	$\mathbb{F}_{2^{131}}$	11 969.93	0.35	13 560	18	-
[2]	$\mathbb{F}_{p_{100}}$	18 720	0.13	500	410.45	under 400
[11]	$\mathbb{F}_{2^{191}}$, $\mathbb{F}_{p_{192}}$	23 000	0.35	68 500	9.89	n.a.
our	$\mathbb{F}_{2^{131}}$	8104*	0.18	500	115	under 30

We underline again that our result for the area complexity does not include RAM. The amount of storage that is required for our implementation is to store $5n$ bits, where n is the number of bits of elements in a field. Assuming factor 6 for each bit of RAM, which is quite conservative, the total area of our processor would be around 12 kgates. This result is close to the result of [5], but only with respect to area. Assuming the same frequency for their processor would result in a latency of almost half a second, which is probably to slow for real applications. The work of Wolkerstorfer is also considering area in mm^2 and power consumption[1] for various technologies. As another comparison our architecture consumes an area smaller than 0.05 mm^2, without RAM.

We can conclude that our architecture presents the smallest known ECC processor for low-cost applications. The performance and power estimates are also implying a feasible solution for various applications of pervasive computing.

6 Conclusions

This work gives a low-power and low footprint processor for ECC suitable for sensor networks. We give detailed results for area and performance estimates for ECC over $\mathbb{F}_{2^p}$ where p is a prime of bit-length varying from 131 to 163. We also include the power numbers obtained by the simulation.

Acknowledgements

Lejla Batina, Nele Mentens and Kazuo Sakiyama are funded by FWO projects (G.0450.04, G.0141.03) and FP6 project SESOC. This research has been also partially supported by the EU IST FP6 project ECRYPT and by IBBT, and K.U. Leuven (OT).

[1] An estimated power is 500 $\mu W/MHz$.

References

1. I. Blake, G. Seroussi, and N. P. Smart. *Elliptic Curves in Cryptography*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.
2. G. Gaubatz, J.-P. Kaps, E. Öztürk, and B. Sunar. State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks. In *2nd IEEE International Workshop on Pervasive Computing and Communication Security (PerSec 2005)*, Kauai Island, Hawaii, March 2005.
3. G. Gaubatz, J.-P. Kaps, and B. Sunar. Public Key Cryptography in Sensor Networks - Revisited. In *1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004)*, Heidelberg, Germany, August 2004.
4. IEEE P1363. Standard specifications for public key cryptography, 1999.
5. S. Kumar and C. Paar. Are standards compliant elliptic curve cryptosystems feasible on RFID? In *Proceedings of Workshop on RFID Security*, Graz, Austria, July 2006.
6. A. Lenstra and E. Verheul. Selecting cryptographic key sizes. In H. Imai and Y. Zheng, editors, *Proceedings of Third International Workshop on Practice and Theory in Public Key Cryptography (PKC 2000)*, number 1751 in Lecture Notes in Computer Science, pages 446–465. Springer-Verlag, 2000.
7. J. López and R. Dahab. Fast multiplication on elliptic curves over GF(2^m). In Ç. K. Koç and C. Paar, editors, *Proceedings of 1st International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 1717 of *Lecture Notes in Computer Science*, pages 316–327. Springer-Verlag, 1999.
8. P. Montgomery. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, Vol. 48:243–264, 1987.
9. C.-P. Schnorr. Efficient Identification and Signatures for Smart Cards. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO '89*, volume LNCS 435, pages 239–252. Springer, 1989.
10. P. Tuyls and L. Batina. RFID-tags for Anti-Counterfeiting. In D. Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006*, Lecture Notes in Computer Science, San Jose, USA, February 13-17 2006. Springer Verlag.
11. J. Wolkerstorfer. Scaling ECC Hardware to a Minimum. In ECRYPT workshop - Cryptographic Advances in Secure Hardware - CRASH 2005, September 6-7 2005. invited talk.