An Elliptic Curve Processor Suitable For RFID Tags

L. Batina[1], J. Guajardo[2], T. Kerins[2], N. Mentens[1], P. Tuyls[2], and I. Verbauwhede[1]

[1] Katholieke Universiteit Leuven, ESAT/COSIC, Belgium
{Lejla.Batina,Nele.Mentens,Ingrid.Verbauwhede}@esat.kuleuven.be
[2] Philips Research Laboratories, Eindhoven, The Netherlands
{Jorge.Guajardo,Tim.Kerins,Pim.Tuyls}@philips.com

Abstract. RFID-Tags are small devices used for identification purposes in many applications nowadays. It is expected that they will enable many new applications and link the physical and the virtual world in the near future. Since the processing power of these devices is low, they are often in the line of fire when their security and privacy is concerned. It is widely believed that devices with such constrained resources can not carry out sufficient cryptographic operations to guarantee security in new applications. In this paper, we show that identification of RFID-Tags can reach high security levels. In particular, we show how secure identification protocols based on the DL problem on elliptic curves are implemented on a constrained device such as an RFID-Tag requiring between 8,500 and 14,000 gates, depending on the implementation characteristics. We investigate the case of elliptic curves over $\mathbb{F}_{2^p}$ with p prime and over composite fields $\mathbb{F}_{2^{2 \cdot p}}$. The implementations in this paper make RFID-Tags suitable for anti-counterfeiting purposes even in the off-line setting.

Key Words: RFID, counterfeiting, authentication, ECC, small area implementations

1 Introduction

RFID-tags are low-cost pervasive devices targeted at providing identification of goods. They consist of an antenna connected to a microchip. Because of the presence of this microchip, they can be considered as the next generation bar codes. One of their main advantages over bar codes is that they can be read out without line of sight. It is expected that in the near future trillions of these devices will be deployed. They will be used to identify goods and provide a link between the physical and the virtual world. It is predicted that this connection will lead to the next revolution after the Internet: The Internet of Things. Currently the main applications for RFID tags include: goods tracking in supply chain management, automated inventory management, automated quality control, access control, payment systems, *etc.* In the future, however, tagged items will also communicate with intelligent devices in the home (intelligent refrigerators, washing machines, *etc.*) and provide additional benefits to consumers. For

example, a refrigerator will automatically detect whether the food is still OK and warn the consumer when necessary, the washing machine will detect the color of clothes in the washing and switch on the appropriate program, and, in general, home appliances will be *intelligent* and be able to communicate with other devices.

An emerging application of RFID-Tags that is being considered is their use for anti-counterfeiting purposes [11]. By locating an RFID-tag with specific product and reference information on a product, one can verify the authenticity of the product. This is done by running a secure protocol between a tag and a reader. If the required information is on the tag and verified to be authentic, the product is declared to be genuine and otherwise not. In a *cloning attack*, the attacker captures the necessary authentication information (obtained *e.g.* by eavesdropping on the channel between the tag and the reader), and stores it in a new chip. In this way the attacker has effectively cloned the original tag. This clone cannot be distinguished from an original tag by a reader. In order to make the cloning of tags infeasible, it should not be possible to derive the tag secrets by active or passive attacks. Recently a lightweight version of such an authentication protocol was developed in [11]. The security of the protocol is based on the Learning Parity in the presence of Noise (LPN) problem. The protocol in [11] is proven secure against passive and against active adversaries in a detection-based model. Reference [25] suggests to use Schnorr's and Okamoto's identification protocols over Elliptic Curves (EC) to provide security against passive and active adversaries, respectively. In addition, [25] provides security against physical attacks as well, thanks to the physical properties of Physically Unclonable Functions (PUFs) [22, 5, 26]. The authors in [25] estimated that ECC and hyperelliptic curve cryptography (HECC) instances of secure identification protocols could be implemented requiring less than 5,000 gates. However, memory requirements were not specified and an explicit construction is not provided.

The fact that tags have very constrained resources (memory, power, speed, area) but need security measures poses very interesting challenges to the security community. First, it is natural to investigate whether existing cryptographic algorithms can be implemented on a tag. Second, it encourages research for new protocols and algorithms targeted at resource constrained devices. Moreover, the research community lacks consensus as to the feasibility of implementing public-key crypto-algorithms on (high-end) RFID tags. For example, [25] claim that public key cryptography on a tag is possible and [1] states: "Unfortunately asymmetric cryptography is too heavy to be implemented on a tag".

1.1 Our Contributions

Feasibility of EC on RFID tags. We address the question of the implementation feasibility of EC based cryptography on a resource constrained device and, in particular, on an RFID-Tag. We answer this question *affirmatively*. We present ECC implementations of secure identification protocols such as Schnorr's [23] on an RFID-Tag. We show that by trading off performance for area it is possible to implement EC public-key cryptography on a tag. Since

area is an important cost factor for the price of RFID-Tags, our main focus is to minimize the area required for the implementations. Our particular implementation of EC over binary fields has an area complexity of between 12,000 and 15,000 equivalent gates depending on the chosen field and implementation. This area complexity includes RAM and assumes a conservative estimate of 6 equivalent gates per RAM cell (i.e. a RAM cell instantiated as a flip-flop). If we were to use dedicated embedded RAM (see for example [19, 9]) our smallest design would require in the order of 8,200 equivalent gates. Notice that by todays standards this corresponds to a mid to high range tag. Although, it is anticipated that in the near future price pressure will continue to limit the number of gates in the ultra low cost tags, it can also be envisioned that eventually this number of gates will be available on all tags.

Our solution is based on identification schemes. We emphasize that our solution is based on identification schemes such as those of Schnorr or Okamoto. This is important because it provided us with an additional way to save area. In contrast, the solution in [27] is based on a challenge-response protocol (CRP) where an ECDSA signature needs to be computed. Such computation, requires the computation of a hash, thus requiring significant hardware resources in addition to the 23,000 equivalent gates of their smallest EC processor design. To our knowledge the best (area optimized) SHA-1 hardware implementation is that of [12], which requires about 4,300 gates[3].

The remainder of the paper is organized as follows. Section 2 gives a brief overview of related work. We present the protocols, EC algorithms and multiplier architectures used in our design in Sect. 3 and 4. In Sect. 5, we describe the processor architecture used for our prototype and estimate the size of the ALU, which is the part of the prototype that contributes the largest area to the overall design. Finally, in Sect. 6 we discuss the results in detail and point out future work.

2 Related Work

Low-power and compact implementations became an important research area with the constant increase in the number of hand-held devices such as mobile phones, smart cards, PDAs *etc.* Schroeppel *et al.* [24] presented a design for ECC over binary fields that was optimized for power, space and time in order to provide digital signatures. The processor in [24] had an area complexity of 191,000 gates. The work of Goodman and Chandrakasan [8] also dealt with energy-efficient solutions. They proposed a domain-specific reconfigurable cryptographic processor (DSRCP) for ECC over both types of finite fields. At 50 MHz, the processor operates at a supply voltage of 2 V and consumes at most

[3] The area estimates of [12] do not include the area for RAM. A similar implementation including the area overhead of RAM requires about 10,000 gates [4].

75 mW of power. In ultra-low-power mode (3 MHz at VDD = 0.7 V), the processor consumes at most 525 μW. Özturk *et al.* [21] introduced modulus scaling techniques that are applicable for ECC over a prime field to develop a low-power elliptic curve processor architecture. They obtained an ECC processor over the 166-bits long prime of size 30,333 gates with the performance of 31.9 msec for point multiplication.

The work of Gaubatz *et al.* [7] discusses the necessity and the feasibility of PKC protocols in sensor networks. In [7], 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 3,000 gates and power consumption of less than 20 μW at 500 KHz. On the other hand, they showed that Rabin is not a feasible solution. In [6] 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 consumes less than 400 μW of power at 500 KHz. The field used was a prime field of order $\approx 2^{100}$.

RFID-based identification is an example of an emerging technology which requires authentication as a cryptographic service. This property can be achieved by symmetric as well as asymmetric primitives. The most notable example of a symmetric cipher implementation is the work of Feldhofer *et al.* [3], which considered implementation of AES on an RFID tag. Recently, Wolkerstorfer [27] showed that ECC based PKC is feasible on RFID-tags by implementing the ECDSA on a small IC. This work is the first complete ECC low-power and compact implementation that meets the constraints imposed by the EPC standard. We compare the implementation of [27] with our results in Section 6 in more detail. We consider here an ECC solution to provide identification for an RFID tag by means of Schnorr's identification scheme as discussed in [25]. If only resistance against passive attacks is needed, the Schnorr Identification scheme can be used as it is known to be secure against passive attacks under the discrete logarithm assumption. An alternative for providing more security is to use Okamoto's identification scheme [20], which is secure against passive, active and concurrent attacks under the DL assumption. Very recently, McLoone and Robshaw [15] reconsider the hardware cost of public-key cryptography for ultra constrained area and power applications. Their implementation allows for a tag that can participate in an authentication protocol a *limited* number of times. As a result, the tag can store in memory the responses to a limited number of authentication and thus, the hardware costs are minimal. In contrast, in this work we consider tags that are able to participate in an authentication protocol an unlimited number of times.

3 Secure Identification Protocols

We investigate the protocol of Schnorr shown in Fig. 1. In this case a tag (prover) proves its identity to a reader (verifier) in a three-pass protocol. As it can be

1. **Common Input:** The set of system parameters in this case consists of: (q, a, b, P, n, h). Here, q specifies the finite field, a, b, define an elliptic curve, P is a point on the curve of order n and h is the cofactor. In the case of tag authentication, most of these parameters are assumed to be fixed.
2. **Prover-Tag Input:** The prover's secret a such that $Z = -a \cdot P$.
3. **Protocol:** The protocol involves exchange of the following messages:

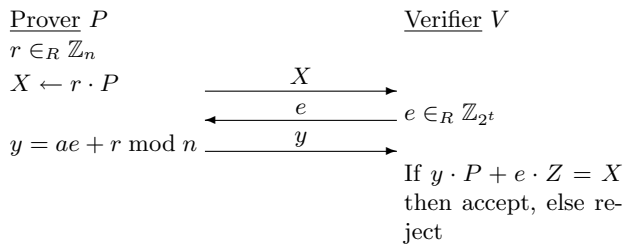

Fig. 1. Schnorr's identification protocol.

observed from the protocol, the critical operation is the point multiplication. Thus, in the remainder of the paper, we describe a processor specifically suited for this operation and cheap enough that it is suitable for anti-counterfeiting RFID applications.

4 ECC implementations for RFID

In this section we elaborate on our choice of algorithms and we explain our strategy to minimize the area of the EC processor. Our strategy can be summarize as follows: (i) we reduce the total number of intermediate registers for calculation of point operations, (ii) we use small digit sizes in our multiplier designs and investigate the effect of a dedicated squarer in the design's area and performance, and (iii) we avoid having to recover the y-coordinate of the elliptic curve point in the tag and, in fact, only operate on the x-coordinate during the protocol. This is, in turn, helps us avoid having to compute two finite field inverses on the tag.

4.1 ECC Operations

In this work, 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 point at infinity, denoted by ∞.

The point or scalar multiplication is the basic operation for cryptographic protocols and it is easily performed via repeated group operations. Here, we describe ECC operations at each level by following the top-down approach. For

the point multiplication we chose Montgomery's [17], which maintains the relationship $P_1 - P_0$ as invariant. Montgomery's method is shown in Algorithm 1. We chose to present Algorithm 1 as in [10], as this provides basic resistance against Simple Power Analysis (SPA) attacks. 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. We chose as starting point for our optimizations the formulae of Lopez and Dahab [14]. The original formulae in [14] require three intermediate registers (two for addition and one for doubling). In our case we eliminate two intermediate registers which added a few more steps to the original algorithms. The result of our optimizations are depicted in Algorithm 2. In this way we made a trade-off between performance and area as point operations require now 6 and 8 multiplications for point addition and doubling (instead of 5 and 6 M), respectively[4].

Algorithm 1 Montgomery's Point Multiplication Method

Require: An integer k and a point $P \in E(\mathbb{F}_q)$
Ensure: $Q = k \cdot P$
1: Set $k \leftarrow (k_{n_k-1} \dots k_1 k_0)_2$
2: Set $P_0 \leftarrow P, \quad P_1 \leftarrow 2 \cdot P$
3: **for** $i = n_k - 2$ downto 0 **do**
4: $\quad b \leftarrow k_i$
5: $\quad P_{1-b} \leftarrow P_0 + P_1, \quad P_b \leftarrow 2 \cdot P_b$
6: **end for**
7: **return** P_0

Algorithm 2 EC point addition and doubling: operations that minimize the number of registers

Require: $X_1, Z_1, X_2, Z_2, x_4 = x(P_2 - P_1)$	**Require:** $b \in \mathbb{F}_{2^n}$, X_1, Z_1
Ensure: $X(P_1 + P_2) = X(P_3) = X_3$, Z_3	**Ensure:** $X(2P_1) = X(P_5) = X_5$, Z_5
1: $Z_3 \leftarrow X_2 \cdot Z_1$	$Z_5 \leftarrow {Z_1}^2$
2: $X_3 \leftarrow X_1 \cdot Z_2$	$Z_5 \leftarrow {Z_5}^2$
3: $Z_3 \leftarrow X_3 + Z_3$	$Z_5 \leftarrow b \cdot Z_5$
4: $Z_3 \leftarrow {Z_3}^2$	$X_5 \leftarrow {X_1}^2$
5: $X_3 \leftarrow X_3 \cdot X_2$	$X_5 \leftarrow {X_5}^2$
6: $X_3 \leftarrow X_3 \cdot Z_1$	$X_5 \leftarrow X_5 + Z_5$
7: $T \leftarrow x_4 \cdot Z_3$	$Z_5 \leftarrow {X_1}^2$
8: $X_3 \leftarrow X_3 + T$	$Z_5 \leftarrow Z_5 \cdot Z_1$
9:	$Z_5 \leftarrow Z_5 \cdot Z_1$

[4] Here, we count squarings also as multiplications.

4.2 $\mathbb{F}_{2^n}$ Arithmetic

Fields of characteristic two in polynomial basis were chosen for this investigation as arithmetic can be implemented efficiently and relatively cheaply in hardware over these fields. Although this is well understood, few previous attempts have been made to develop truly low area implementations of this arithmetic for ECC. Addition of two elements $c = a + b \in \mathbb{F}_{2^n}$ is performed via an n–bitwise logical XOR operation. The standard way to compute the product $c = a \cdot b \in \mathbb{F}_{2^n} \cong \mathbb{F}_2[x]/f(x)$, and $a = \sum_{i=0}^{n-1} a_i x^i$, $b = \sum_{j=0}^{n-1} b_j x^j$, $f = x^n + \sum_{i=0}^{s} f_i x^i$, $s < n$, is given by

$$c = \sum_{j=0}^{n-1} \sum_{i=0}^{n-1} a_i b_j x^{i+j} \bmod f = a \sum_{j=0}^{n-1} b_j x^j \bmod f \tag{1}$$

This represents the most compact solution, where the $b_j a x^j$ partial products from (1) are computed iteratively and reduction modulo f of the degree n partial product polynomial is performed on each of the n iterations. The digit serial multiplication algorithm [13] may be considered as a generalization of this. Rather than processing the binary coefficients b_j of $b \in \mathbb{F}_{2^n}$ serially, a number of them are processed in parallel. Here there is scope to trade-off an increase in gate count for increased performance. This is an important consideration in low frequency implementations over relatively small (composite) fields as discussed here.

Here $b = \sum_{j=0}^{n-1} b_j x^j$, rather than being considered as n coefficients of $\mathbb{F}_2$ is considered as being composed of $d = \lceil \frac{n}{D} \rceil$ *words*, each word containing D elements of $\mathbb{F}_2$. Now $b = \sum_{k=0}^{d-1} \tilde{b}_k x^{kD}$, each $\tilde{b}_k = \sum_{l=0}^{D-1} b_{l+kD} x^l$, and

$$c = \sum_{k=0}^{d-1} (a\tilde{b_k}) x^{kD} \bmod f \tag{2}$$

can be calculated in d iterations. Notice that the $a\tilde{b}_k$ partial products are calculated recursively. A variant of the Song-Parhi method is illustrated as Algorithm 3. When $D = 1$ then $d = n$ and $\tilde{b}_k = b_j \in \mathbb{F}_2$ and this method reverts to Horner multiplication. Squaring $c = a^2 \in \mathbb{F}_{2^n}$ is a special case of multiplication. It is well known that $a^2 = \sum_{i=0}^{n-1} a_i x^{2i}$, which can then be reduced modulo f to a field element in $\mathbb{F}_{2^n}$.

For security reasons it is typically recommended to use fields $\mathbb{F}_{2^p}$ where p is a prime. As an example we investigate the cases where $p = 131$ and $p = 139$. However, we also consider EC over a quadratic extension of $\mathbb{F}_{2^p}$. For example, we consider $\mathbb{F}_{2^{134}} \equiv \mathbb{F}_{(2^{67})^2} \equiv \mathbb{F}_{2^{67}}[y]/g(y)$, where $\deg(g) = 2$ and g is an irreducible polynomial over $\mathbb{F}_{2^{67}}$. In this way we can translate the arithmetic from $\mathbb{F}_{(2^p)^2}$ to $\mathbb{F}_{2^p}$, which results in a reduction in the size of ALU by a factor of two approximately. In a composite field $\mathbb{F}_{(2^p)^2}$ each element can be represented as $z = xt + y$ where $x, y \in \mathbb{F}_{2^p}$. The security implied by this choice of fields (lower bit-length than standardized counterparts possibly combined with composite degree) is discussed in the extended version of this paper [2].

Algorithm 3 Digit serial multiplication in $\mathbb{F}_{2^n}$

Require: $a = \sum_{i=0}^{n-1} a_i x^i$, $b = \sum_{k=0}^{d-1} \tilde{b_k} x^{kD}$ where $\tilde{b_k} = \sum_{l=0}^{D-1} b_{l+kD} x^l$ and $f \in \mathbb{F}_2[x]$
Ensure: $c = a \cdot b \bmod f(x)$
1: $c \leftarrow 0$
2: **for** k from 1 to $d-1$ **do**
3: $\quad c \leftarrow x^D(c + \tilde{b}_{d-1}a) \bmod f$
4: $\quad b \leftarrow x^D b$ {Only a D-bit left shift}
5: **end for**
6: $c \leftarrow (c + \tilde{b}_{d-1}a) \bmod f$
7: Return c

4.3 Recovering the y coordinate of $Q = k \cdot P$

In traditional solutions, after computing $Q = k \cdot P$, one is required to transform back to affine coordinates and compute the y-coordinate of Q. We, however, advocate a different solution. One simple solution is to send both the end values of registers containing P_1 and P_2 in Algorithm 1 to the verifier so that the verifier himself can recover the y-coordinate of Q. This would incur in the sending of four finite field elements, corresponding to the projective coordinate representation of P_1 and P_2. Alternatively, the protocol can be run by only using the x-coordinates of all points involved. Notice that this was first observed by Miller in his seminal paper [16]. In either case, the projective coordinates sent to the verifier should be masked with a random value to avoid the attack described in [18]. This requires two extra multiplications at the end of the point multiplication which is negligible in comparison to the rest of the computation.

5 Elliptic Curve Processor Architecture

Our Elliptic Curve Processor (ECP) for RFID is shown in Fig. 2. The operational blocks are as follows: a Control Unit(CU), an Arithmetic Logic Unit (ALU), and Memory (RAM and ROM). In ROM the ECC parameters and the constants x_4 (the x-coordinate of $P_2 - P_1$) 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 CU controls scalar multiplication and point operations. In the case of composite fields implementations, it also controls the operations in extension fields. In addition, the controller commands the ALU which performs field multiplication, addition and squaring. The bits of $k = \sum_{i=0}^{n_k-1} k_i 2^i$, $k_i \in \{0,1\}$, $n_k = \lceil \log_2 k \rceil$, 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. This is processed in an n-bit shift register. 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 seven n-bit ($n = p$) registers for ordinary fields and nine n-bit ($n = 2p$) registers for composite fields. The extra registers are required to store intermediate values when performing multiplication in composite fields. Table 1

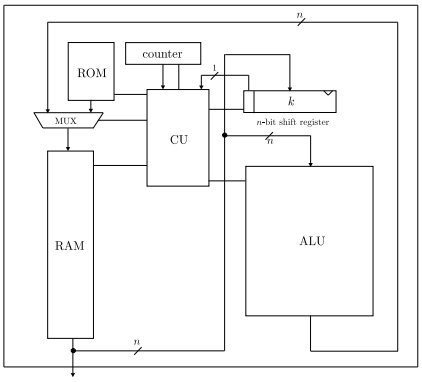

Fig. 2. ECP Architecture.

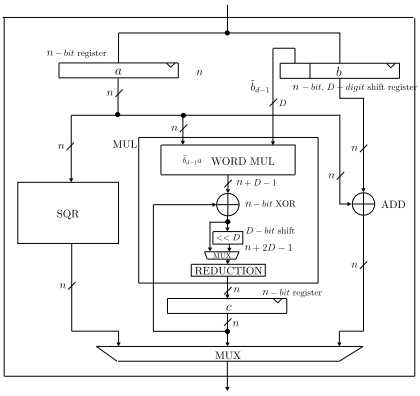

Fig. 3. ALU Architecture.

summarizes the number of cycles required for basic operations and for a whole point multiplication in an EC over $\mathbb{F}_{2^p}$.

Table 1. Cycle count for basic arithmetic operations and EC operations over $\mathbb{F}_{2^p}$. L: Load, C: Computation, S: Store

Operation	L	C	S	Total Cycles
$\mathbb{F}_{2^p}$ addition	2	1	1	4
$\mathbb{F}_{2^p}$ squaring	1	1	1	3
$\mathbb{F}_{2^p}$ multiplication	2	$\lceil \frac{p-1}{D} \rceil$	1	$\lceil \frac{p-1}{D} \rceil + 3$
EC operations assuming a squarer				
EC addition ($\mathbb{F}_{2^p}$)	$5\ MUL + 1\ SQ + 2\ ADD = 5\lceil \frac{p-1}{D} \rceil + 26$			
EC double ($\mathbb{F}_{2^p}$)	$3\ MUL + 5\ SQ + 1\ ADD = 3\lceil \frac{p-1}{D} \rceil + 27$			
Point mult. ($\mathbb{F}_{2^p}$)	$(n_k - 1)\left(8\lceil \frac{p-1}{D} \rceil + 53\right)$			
EC operations assuming no squarer				
EC addition ($\mathbb{F}_{2^p}$)	$6\ MUL + 2\ ADD = 6\lceil \frac{p-1}{D} \rceil + 26$			
EC double ($\mathbb{F}_{2^p}$)	$8\ MUL + 1\ ADD = 8\lceil \frac{p-1}{D} \rceil + 28$			
Point mult. ($\mathbb{F}_{2^p}$)	$(n_k - 1)\left(14\lceil \frac{p-1}{D} \rceil + 54\right)$			

5.1 The Arithmetic Logic Unit

The largest contribution in area to the overall design comes from the ALU, illustrated in Fig. 3. It consists of two n-bit registers a and c, and an n-bit shift-register b that outputs D bits at the time. In addition, the ALU has circuitry for implementing addition, squaring and multiplication in $\mathbb{F}_{2^n}$. Load and store operations between the ALU and memory cost a single clock cycle. The ADD block consists of n XOR gates, and the SQR block consists of at most $\frac{3n}{2}$ XOR gates and computes $\mathbb{F}_{2^n}$ additions and squarings in a single clock cycle once data has been loaded into the ALU. The MUL block implements an iteration of Step 3 of Algorithm 3 in a single clock cycle. Multiplication is calculated then in $d = \lceil \frac{n}{D} \rceil$ clock cycles. For composite fields, the field arithmetic translates to the

arithmetic in the subfield as follows: (i) addition in $\mathbb{F}_{(2^p)^2}$ requires 2 additions in $\mathbb{F}_{2^p}$, (ii) multiplication in $\mathbb{F}_{(2^p)^2}$ requires 3 multiplications and 4 additions in $\mathbb{F}_{2^p}$, and (iii) squaring in $\mathbb{F}_{(2^p)^2}$ requires 2 squaring and 1 addition in $\mathbb{F}_{2^p}$.

Remark 1. Our processor does not include at this moment functionality to compute $ae+r \bmod n$ in Schnorr's protocol. We expect such functionality to require no more than 1000 additional gates.

5.2 A Word Regarding Power

At the present moment, we do not have an actual chip and we lack explicit power measurements for our simulations. Nevertheless, we believe that attaining the power values required for RFID applications using our design is possible. In fact, our processor architecture is very similar to the architecture presented in [27]. One particular characteristic of both designs is the usage of an Arithmetic Logic Unit (ALU) with a full-precision data path. The differences are on the details of our implementation: field size, choice of finite field arithmetic methodology (Montgomery vs. dedicated trinomial or pentanomial circuits), support for multiple fields versus support for a single field, hashing versus no hashing required in our implementation, etc. In general, our design is aimed at making our implementation as specific as possible to our particular application. This methodology leads to significant complexity reduction in the area requirements and thus also to power savings. Thus, since [27] was able to attain the power requirements of an RFID system, we are confident that our design, being smaller and simpler, will also attain the required power figures at the same or lower operating frequencies.

6 Results and discussion

In this section, we provide estimates for the latency and the area complexity of Schnorr's protocol. As mentioned above the core part of the protocol is one point multiplication. The results for various architectures are given in Tables 2 and 3. We considered solutions with or without the squarer as it allows also for a trade-off between area and performance. For the case of composite fields the ALU shrinks in size but some speed-up is then necessary which we obtain by means of a digit-serial multiplier (instead of a bit-serial one, *i.e.,* $D = 1$). The performance in each case is calculated by use of formulae for point operations as in Algorithm 2 and we calculate the total number of cycles for each case assuming the numbers for field arithmetic provided in Sect. 5. The designs were synthesized using Synopsis Design-analyzer for the frequency of 175 kHz and a 0.25 μm CMOS library. One of our main reasons for using composite fields was to reduce the ALU's area. This is clearly visible in Tables 2 and 3. We notice that the ALU varies in size from 2,863 to 7,379 gates and the smallest one is obtained for the field $\mathbb{F}_{(2^{67})^2}$, without the squarer and with a bit-serial multiplier. However, the performance is the worst for this case, requiring more than 2 seconds for

Table 2. Implementation results @ 175 kHz and assuming a dedicated squarer circuit.

Implementation		ALU	RAM	Perf. @175 kHz	Area wo RAM	AT factor	AT f.
Digit size	Field Type	[kgates]	[bits]	[s]	[kgates]	[wo. RAM]	[w. RAM]
D=1	$\mathbb{F}_{2^{131}}$	6306	917	0.81	8582	6975	11446
	$\mathbb{F}_{(2^{67})^2}$	3274	1206	1.44	6074	8734	19139
	$\mathbb{F}_{2^{139}}$	6690	973	0.91	9044	8259	13590
D=2	$\mathbb{F}_{2^{131}}$	6962	917	0.43	9233	3937	6284
	$\mathbb{F}_{(2^{67})^2}$	3610	1206	0.84	6410	5359	11409
	$\mathbb{F}_{2^{139}}$	7379	973	0.48	9734	4652	7442
	$\mathbb{F}_{(2^{71})^2}$	3648	1278	0.92	6534	6044	13137
D=3	$\mathbb{F}_{(2^{67})^2}$	3789	1206	0.64	6589	4187	8784
	$\mathbb{F}_{(2^{71})^2}$	3833	1278	0.71	6719	4786	10248
D=4	$\mathbb{F}_{(2^{67})^2}$	4103	1206	0.54	6903	3757	7694
	$\mathbb{F}_{(2^{71})^2}$	4152	1278	0.60	7038	4197	8769

Table 3. Implementation results @ 175 kHz and assuming no dedicated squarer circuit.

Implementation		ALU	RAM	Perf. @175 kHz	Area wo RAM	AT factor	AT f.
Digit size	Field Type	[kgates]	[bits]	[s]	[kgates]	[wo RAM]	[wRAM]
D=1	$\mathbb{F}_{2^{131}}$	5679	917	1.39	7953	11072	18731
	$\mathbb{F}_{(2^{67})^2}$	2953	1206	2.39	5708	13648	30949
	$\mathbb{F}_{2^{139}}$	6018	973	1.57	8380	13124	22267
D=2	$\mathbb{F}_{2^{131}}$	6335	917	0.72	8603	6161	10101
	$\mathbb{F}_{(2^{67})^2}$	3289	1206	1.34	6044	8085	17764
	$\mathbb{F}_{2^{139}}$	6718	973	0.80	9079	7303	11999
	$\mathbb{F}_{(2^{71})^2}$	3463	1278	1.49	6304	9367	20759
D=3	$\mathbb{F}_{(2^{67})^2}$	3468	1206	0.99	6224	6140	13279
	$\mathbb{F}_{(2^{71})^2}$	3647	1278	1.11	6489	7226	15764
D=4	$\mathbb{F}_{(2^{67})^2}$	3782	1206	0.83	6537	5406	11389
	$\mathbb{F}_{(2^{71})^2}$	3967	1278	0.91	6808	6199	13180

one point multiplication. The total area without RAM includes the sizes of the ALU, the CU, the counter and the shift-register. The largest portion of that is occupied by the key register *i.e.* 1,400 and 1,500 gates for fields $\mathbb{F}_{2^{131}}$ and $\mathbb{F}_{2^{139}}$, respectively. The control logic takes between 10% and 15% of a whole design.

In the last two columns, we computed the area-time product for two cases, including RAM and not including RAM. To map the number of bits to be stored to actual gates we used a factor of 6, which is conservative when using SRAM. If we were to use dedicated embedded RAM, it would be possible to half the area requirement (see for example [19, 9]), at the very least. From Table 2 and looking at the AT-product values, we conclude that, in general, it is beneficial to use a digit-serial multiplier and a squarer. However, these options are not the most compact. For compactness one should choose an implementation without squarer. The total area is expressed without RAM for two reasons. First, it is hard to exactly map it to the corresponding number of gates and second, most tags have RAM available. Some high-end tags have therefore the possibility to store 1,000 bits, which would be enough in some cases presented in Table 2. We compare our results with other related work in Table 4. It is hard to compare with other related work as there is no previous ECC implementation suitable for RFIDs. We chose here the architecture with the best timing although it is possible to have an adequate solution requiring a total of 13,646 gates with a

performance that is still below 1 second (0.84 sec). We stress here again that we obtain these figures by including very conservative estimates for RAM in the total gate count. In fact, a RAM cell that requires 6 equivalent gates to be implemented is a register cell. A typical full-custom RAM cell requires somewhere between 1 and 2 equivalent gates, thus bringing the total area required for the design under 10,000 gates. Other optimizations involve the shift register for the key, which is of full length and requires 1,500 gates. This can still be improved by loading the key in two or more parts, thus, reducing the area significantly.

Table 4. Performance and area of different algorithms and implementations

Source	Algorithm	Finite field/ Parameter Size	Area $[gates]$	Technology $[\mu m]$	Op. Frequency [kHz]	Performance $[ms]$
[6]	NTRUEncrypt	$N = 167, p = 3, q = 128$	3000	0.13	500	58.45
[3]	AES	block size = 128 bits	3595	0.35	100	10.2 (1016 cycles)
[12]	SHA-1	data size = 512 bits	4276	0.13	500	0.81 (405 cycles)
this work (smallest area)	EC	$\mathbb{F}_{(2^{67})^2}$	12,944	0.25	175	2.39 sec.
this work (smallest AT product, fastest)	EC	$\mathbb{F}_{2^{131}}$	14,735	0.25	175	430
[6]	EC	$\mathbb{F}_{p_{100}}$	18,720	0.13	500	410.5
[27]	EC	$\mathbb{F}_{2^{191}}$ and $\mathbb{F}_{p_{192}}$	23,000	0.35	68,500	6.7
[21]	EC	$\mathbb{F}_{p_{166}}$	30,333	0.13	20,000	31.9

7 Concluding Remarks

This work provides evidence that ECC on RFID might be a viable solution in the near future. This is important as it allows much more sophisticated protocols based on public-key cryptography than currently being considered for use in RFID. We investigated several options considering ECC over $\mathbb{F}_{2^p}$, p a prime, operands ranging between 130 and 140 bits in length, and composite fields. We also considered different ALU configurations to obtain more compact and still acceptable performance. We follow design criteria that would lead to low-power implementations, *i.e.* we try to minimize the area and reduce the operating frequency. The best architecture with respect to both area and performance is slightly larger than 10,000 gates. Future work will investigate the exact amount of power consumed by our processors, the cost of side-channel attack countermeasures, and concentrate on the further investigation of protocols based on public-key cryptography for RFID.

References

1. G. Avoine, E. Dysli, and P. Oechslin. Reducing Time Complexity in RFID Systems. In B. Preneel and S. E. Tavares, editors, *Selected Areas in Cryptography — SAC 2005*, volume 3897 of *LNCS*, pages 291–306. Springer, 2005.

2. L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede. An Elliptic Curve Processor Suitable For RFID-Tags. Cryptology ePrint Archive, Report 2006/227, July 4th, 2006. Available at `http://eprint.iacr.org/`.
3. M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong Authentication for RFID Systems using the AES Algorithm. In M. Joye and J. J. Quisquater, editors, *Cryptographic Hardware in Embedded Systems — CHES 2004*, volume 3156 of *LNCS*, pages 357–370. Springer-Verlag, 2004.
4. M. Feldhofer and C. Rechberger. A case against currently used hash functions in RFID protocols. Printed handout of Workshop on RFID Security – RFIDSec 06, pages 109–122. ECRYPT Network of Excellence, July 2006. Available at `http://events.iaik.tugraz.at/RFIDSec06/Program/index.htm`.
5. B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas. Silicon physical unknown functions. In V. Atluri, editor, *ACM Conference on Computer and Communications Security — CCS 2002*, pages 148–160. ACM, November 2002.
6. 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.
7. G. Gaubatz, J.-P. Kaps, and B. Sunar. Public Key Cryptography in Sensor Networks - Revisited. In C. Castelluccia, H. Hartenstein, C. Paar, and D. Westhoff, editors, *Security in Ad-hoc and Sensor Networks — ESAS 2004. Revised Selected Papers*, volume 3313 of *LNCS*, pages 2–18. Springer, 2004.
8. J. Goodman and A.P. Chandrakasan. An energy-efficient reconfigurable public-key cryptography processor. *IEEE Journal of Solid-State Circuits*, 36(11):1808–1820, November 2001.
9. K. Itoh. Low-Voltage Embedded RAMs in the Nanometer Era. In *IEEE International Conference on Integrated Circuits and Technology — ICICT 2005*, pages 235–242. IEEE Computer Society, 2005.
10. M. Joye. Elliptic Curves and Side-Channel Analysis. *ST Journal of System Research*, 4(1):283306, 2003.
11. A. Juels and S. A. Weis. Authenticating pervasive devices with human protocols. In V. Shoup, editor, *Advances in Cryptology — CRYPTO 2005*, volume 3621 of *LNCS*, pages 293–308. Springer-Verlag, 2005.
12. J.-P. Kaps and B. Sunar. Energy comparison of AES and SHA-1 for ubiquitous computing. In E. Sha, editor, *IFIP International Conference on Embedded and Ubiquitous Computing — EUC 2006*, LNCS. Springer, 2006. To appear.
13. L. Song and K.K. Parhi. Low Energy Digit-Serial/Parallell Finite Field Multipliers. *Kluwer Journal of VLSI Signal Processing Systems*, 19(2):149–166, 1998.
14. J. López and R. Dahab. Fast multiplication on elliptic curves over GF(2^m). In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES*, volume 1717 of *LNCS*, pages 316–327. Springer-Verlag, 1999.
15. M. McLoone and M. J. B. Robshaw. Public Key Cryptography and RFID Tags. In M. Abe, editor, *Topics in Cryptology — CT-RSA 2007*, LNCS. Springer, 2007. To appear.
16. V. S. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology – CRYPTO '85*, volume 0218 of *LNCS*, pages 417–426. Springer-Verlag, 1986.
17. P. Montgomery. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, 48(177):243–264, 1987.

18. D. Naccache, N. P. Smart, and J. Stern. Projective coordinates leak. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 257–267. Springer, 2004.
19. Y. Nakagome, M. Horiguchi, T. Kawahara, and K. Itoh. Review and future prospects of low-voltage RAM circuits. *IBM Journal of Research and Development*, 47(5/6):525–552, 2003.
20. T. Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, 1992.
21. E. Özturk, B. Sunar, and E. Savaş. Low-Power Elliptic Curve Cryptography Using Scaled Modular Arithmetic. In M. Joye and J. J. Quisquater, editors, *Cryptographic Hardware in Embedded Systems — CHES 2004*, volume 3156 of *LNCS*, pages 92–106. Springer-Verlag, 2004.
22. R. Pappu. Physical one-way functions. *Science*, 297(6):2026, 2002.
23. C.-P. Schnorr. Efficient Identification and Signatures for Smart Cards. In G. Brassard, editor, *Advances in Cryptology — CRYPTO '89*, volume 435 of *LNCS*, pages 239–252. Springer, 1989.
24. R. Schroeppel, C. L. Beaver, R. Gonzales, R. Miller, and T. Draelos. A low-power design for an elliptic curve digital signature chip. In B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2002*, volume 2523 of *LNCS*, pages 366–380, 2002.
25. P. Tuyls and L. Batina. RFID-tags for Anti-Counterfeiting. In D. Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006*, LNCS, San Jose, USA, February 13-17 2006. Springer Verlag.
26. P. Tuyls and B. Skoric. Secret Key Generation from Classical Physics: Physical Uncloneable Functions. In S. Mukherjee, E. Aarts, R. Roovers, F. Widdershoven, and M. Ouwerkerk, editors, *AmIware: Hardware Technology Drivers of Ambient Intelligence*, volume 5 of *Philips Research Book Series*. Springer-Verlag, September 2006.
27. J. Wolkerstorfer. Scaling ECC Hardware to a Minimum. In ECRYPT workshop - Cryptographic Advances in Secure Hardware - CRASH 2005, September 6-7 2005. Invited talk.