Side-channel aware design: Algorithms and Architectures for Elliptic Curve Cryptography over $\text{GF}(2^n)$

Lejla Batina, Nele Mentens, Bart Preneel, Ingrid Verbauwhede
Katholieke Universiteit Leuven, ESAT/SCD-COSIC

{Lejla.Batina, Nele.Mentens, Bart.Preneel, Ingrid.Verbauwhede}@esat.kuleuven.ac.be

Abstract

This paper proposes efficient algorithms for Elliptic Curve Cryptography (ECC). As an example a compact and efficient FPGA architecture for ECC over finite fields of even characteristic is presented. The implementation is balanced in order to increase the security w.r.t. simple side-channel attacks.

Keywords: Multiplication in $\text{GF}(2^n)$, Hardware implementation, Systolic array architecture, Elliptic Curve Cryptography (ECC), Montgomery method for point multiplication

1 Introduction

The best-known and most commonly used public-key cryptosystems (PKC) are based on factoring (RSA) and on the discrete logarithm problem (Diffie-Hellman, DSA) [19]. They allow secure communications over insecure channels without prior agreement of a shared secret and they also enable digital signatures. Another alternative for PKC is Elliptic Curve Cryptography (ECC), which was proposed in the mid 1980s by Miller [21] and Koblitz [14].

In this article we propose an implementation of the approach of Montgomery [22] for scalar multiplication in binary fields. It uses a representation where computations are performed on the x -coordinate only. Menezes and Vanstone observed that the benefit in storage comes at a considerable expense of speed [20]. Also, from an algorithmic point of view, Stam concludes that it is less efficient than other known methods and that in the binary case it can hardly be recommended [26]. However, our conclusion is just the opposite, at least for hardware implementations. This method can benefit from independent calculations for point operations that can be therefore performed fully in parallel by means of two multipliers. Furthermore, we have optimized the

formulae for the point operations to have exactly the same number of field multiplications for point addition and doubling. The field multiplications are performed in corresponding steps in both point operations. We are convinced that this approach also offers an improved resistance against side-channel attacks compared to other unbalanced methods.

The remainder of this paper is organized as follows. Section 2 provides the necessary mathematical background. In Sect. 3 previous work is discussed. Section 4 gives algorithms and details of the new implementation. In Sect. 5 the results of our FPGA implementation of the ECC processor are presented including a comparison with other relevant work. Section 6 addresses the security w.r.t. simple side-channel attacks. It also includes graphs of power consumption which prove improved side-channel resistance. Section 7 concludes the paper.

2 Elliptic Curves over $\text{GF}(2^n)$

The point or scalar multiplication is the basic operation for cryptographic protocols; it is easily performed via repeated group operations. At the next (lower) level are the point operations, which are closely related to the coordinates used to represent the points. The lowest level consists of finite field operations such as addition, subtraction, multiplication and inversion required to perform the group operations.

We introduce some notation. Let $P_4 = (x_4, y_4) = P_2 - P_1$ and $P_5 = (x_5, y_5) = 2P_1$ with $P_3 = P_1 + P_2$. The point P_4 is included because the method for point multiplication, as introduced by Montgomery, is defined by the fact that to add two points their difference should be known (while y -coordinate is not needed). The formulae for point addition and doubling from [4] can be rewritten using that $P_1 = (x_1, y_1) \in E$. For $P_1 \neq P_2$ we get:

$$\begin{aligned} x_3 &= \left(\frac{y_1 + y_2}{x_1 + x_2} \right)^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a, \\ y_3 &= \left(\frac{y_1 + y_2}{x_1 + x_2} \right)(x_1 + x_3) + x_3 + y_1. \end{aligned} \quad (1)$$

If $P_1 = P_2$,

$$\begin{aligned} x_5 &= x_1^2 + \frac{b}{x_1^2} \\ y_5 &= x_1^2 + (x_1 + \frac{y_1}{x_1})x_5 + x_5. \end{aligned}$$

For P_4 we get from Blake *et al.* [4, Lemma III.2]:

$$x_4 = \left(\frac{y_1 + y_2 + x_2}{x_1 + x_2} \right)^2 + \frac{y_1 + y_2 + x_2}{x_1 + x_2} + x_1 + x_2 + a.$$

We will use the observation that the x -coordinate of P_5 does not include the y -coordinate of P_1 . Also the x -coordinate of the sum of P_3 and P_4 can be expressed with the x -coordinate only. More precisely, we have:

Lemma 2.1 *The x -coordinates of the points $P_3 = (x_3, y_3) = P_1 + P_2$ and $P_4 = (x_4, y_4) = P_2 - P_1$ on an elliptic curve (1) satisfy:*

$$x_3 + x_4 = \frac{x_1 \cdot x_2}{(x_1 + x_2)^2}$$

Proof: It follows directly from the formula for addition and the curve equation.

3 Previous Work on Hardware Implementations of ECC

This section lists some relevant previous work on ECC architectures for binary fields. There are many papers [23, 9, 10, 12] dealing with this topic but very few efficient hardware implementations present a completely generic solution which allows an arbitrary choice for all parameters. Orlando and Paar [23] proposed a scalable elliptic curve processor architecture which operates over finite fields $\text{GF}(2^m)$. Goodman and Chandrakasan proposed a cryptographic processor [9], which performs a variety of algorithms for PKC applications. Gura *et al.* [10] have introduced a programmable hardware accelerator for ECC over $\text{GF}(2^n)$.

The first bit-serial multiplier was discussed by Beth and Gollmann [3]. This multiplier uses convolution and reduction modulo an irreducible polynomial and takes n clock cycles to compute a multiplication. Relevant algorithms and architectures for multiplication in $\text{GF}(2^n)$ have been proposed in [6, 3]. For a detailed survey on finite fields multipliers and processors for PKC see Batina *et al.* [2].

4 A New Hardware Implementation

In this section we describe our new hardware implementation. We follow the top-down approach and for each step we elaborate our choice.

4.1 Montgomery Method for Point Multiplication in $\text{GF}(2^n)$

For the point multiplication we chose the method of Montgomery [22]. The algorithm used (Algorithm 1) was considered by López and Dahab [18].

Algorithm 1 Algorithm for point multiplication

Require: an integer $k > 0$ and a point P

Ensure: $x(kP)$

```

1:  $k \leftarrow k_{l-1}, \dots, k_1, k_0$ 
2:  $P_1 \leftarrow P, P_2 \leftarrow 2P$ 
3: for  $i$  from  $l-2$  downto 0 do
4:   If  $k_i = 1$  then
5:      $x(P_1) \leftarrow x(P_1 + P_2), x(P_2) \leftarrow x(2P_2)$ 
6:   Else
7:      $x(P_2) \leftarrow x(P_1 + P_2), x(P_1) \leftarrow x(2P_1)$ 
8:   end for
9: Return  $x(P_1)$ 
```

The advantage of this algorithm is that it calculates one point addition and one doubling in each step. Moreover, the algorithm requires less registers compared to other hardware solutions. This could be of interest for implementations in constrained environments.

4.2 Point Addition and Doubling

At this level our design is improved with respect to other proposals. Namely, point operations (add and double) are in principle different, which can be explored from the point of side-channel analysis. Some authors have tried before to balance these two operations in order to improve side-channel resistance. We mention here the work of Brier and Joye [5] who suggested two approaches to achieve uniformity of point operations. However, both approaches result with some penalty in speed.

In the formulae of López and Dahab in $\text{GF}(2^n)$ point operations are almost balanced as they have $A : D = 5M : 6M$. Here, A and D are the point operations and M is a field multiplication. Consider the formulae for point operations in the case of simple projective coordinates *i.e.* $x_i = (X_i/Z_i), i = 1, 2$.

The results of point doubling and point addition, i.e. $X_5 = X(P_5)$ and $X_3 = X(P_3) = X(P_1 + P_2)$ respectively, are calculated as:

$$\begin{aligned} X_5 &= X_1^4 + bZ_1^4 \\ Z_5 &= X_1^2 \cdot Z_1^2 \\ X_3 &= (X_1 \cdot Z_2 + X_2 \cdot Z_1)^2 \\ Y_3 &= x_4Z_3 + (X_1 \cdot X_2) \cdot (Z_1 \cdot Z_2) \end{aligned} \quad (2)$$

It is easy to see that point doubling and addition would require 6 and 5 multiplications respectively. We slightly rewrote the formulae in order to have 6 multiplications for both point operations. We had to add one more multiplication in the point addition, so we used the following formula:

$$X_1Z_2 + X_2Z_1 = (X_1 + X_2)(Z_1 + Z_2) - X_1Z_1 - X_2Z_2,$$

that follows from the Karatsuba-like approach [13]. The algorithms for point addition and doubling are given in Algorithm 2.

Algorithm 2 EC point addition and doubling

Require: $a, c \in \text{GF}(2^n)$, $c = b^{2^{m-1}}$, X_i, Z_i
for $i = 1, \dots, 4$, $x_i = \frac{X_i}{Z_i}$, $x_4 = x(P_1 - P_2)$

Ensure: $X(P_1 + P_2) = X(P_3) = X_3$

1. $T_1 \leftarrow x_4, X_3 \leftarrow X_1 + X_2, Z_3 \leftarrow Z_1 + Z_2$
2. $Z_3 \leftarrow X_3 \cdot Z_3$
3. $T_2 \leftarrow X_1Z_1$
4. $X_3 \leftarrow X_2Z_2$
5. $Z_3 \leftarrow Z_3 - T_2 - X_3$
6. $Z_3 \leftarrow Z_3^2$
7. $T_1 \leftarrow x_4Z_3$
8. $X_3 \leftarrow T_2X_3$
9. $X_3 \leftarrow X_3 + T_1$

Require: $a, c \in \text{GF}(2^n)$, $c = b^{2^{m-1}}$,
 X_1, Z_1 where $x_1 = \frac{X_1}{Z_1}$

Ensure: $X(2P_1) = X(P_5) = X_5$

1. $T_1 \leftarrow c$
 2. $Z_5 \leftarrow Z_1^2$
 3. $X_5 \leftarrow X_1^2$
 4. $T_1 \leftarrow Z_5T_1$
 5. $T_1 \leftarrow T_1 + X_1 - X_1$
 6. $Z_5 \leftarrow X_5Z_5$
 7. $T_1 \leftarrow T_1^2$
 8. $X_5 \leftarrow X_5^2$
 9. $X_5 \leftarrow X_5 + T_1$
-

Each point operation requires exactly 6 multiplications which are also balanced. In Step 5 of the point doubling a redundant operation is inserted to balance even the field additions. The required number of registers is 3 for both cases. More precisely, the following lemma holds:

Lemma 4.1 *When Algorithm 1 deploys algorithm 2 we get the following number of operations in $\text{GF}(2^n)$:*

$$\begin{aligned} \#inversions &= 1 \\ \#multiplications &= 12\lfloor \log_2 k \rfloor + 13 \\ \#additions &= 6\lfloor \log_2 k \rfloor + 7 \end{aligned}$$

Note that the 12 multiplications require only the time for 6 multiplications since two multiplications are performed in parallel in every iteration of the main loop.

4.3 An Algorithm for Field Multiplication

Our circuit implements Algorithm 3; it includes two parts, classical and Montgomery, each of which is a systolic array. Those two arrays process the operand $a(x)$ from different sides and they stop after exactly $\lceil n/2 \rceil$ cycles for the bit-serial version and after $\lceil s/2 \rceil$ for this new digit-serial architecture. In short, let us denote $a_{MSB}(x)$ and $a_{LSB}(x)$ as the most significant and the least significant half of $a(x)$, respectively. After exactly $\lceil s/2 \rceil$ steps the classical and the Montgomery part have calculated $a_{MSB}(x) \cdot b(x) \cdot x^{\lceil s/2 \rceil w}$ and $a_{LSB}(x) \cdot b(x) \cdot x^{-\lceil s/2 \rceil w}$, respectively. So each part evaluated half of the polynomial $a(x)$ and XOR-ing them will give the M-residue of the multiplication result $Res(x)$ with $r(x) = x^{\lceil s/2 \rceil w}$. After conversion from Montgomery to the normal representation, we get the result. Namely, the following lemma holds.

Lemma 4.2 *The result of Algorithm 3 is in the Montgomery domain i.e. $Res(x) = a(x)b(x)r^{-1}(x)$ where $r(x) = x^{\lceil \frac{n}{2} \rceil}$.*

The idea of combining two algorithms together was mentioned in [24] and the schematic of the multiplier was presented in [1].

In Algorithm 3, $A_i(x)$ represents one digit of the polynomial $a(x)$. Also, here $Res_{M_0}(x)$ and $F_0(x)$ are the least significant words of $Res_M(x)$ and $f(x)$ respectively. On the other hand, $Res_{C_{n-1}}(x)$ corresponds to the most significant word of $Res_C(x)$.

4.4 A prototype FPGA architecture

Our Elliptic Curve Processor (ECP) is shown in Fig. 1. The operation blocks on each level from top to bottom are as follows:

- Level 1: Main Controller

Algorithm 3 Digit-serial Modular Multiplication in $GF(2^n)$

Require: polynomials $a(x)$, $b(x)$ and $f(x)$, $F_0'(x)$

Ensure: $Res(x) = a(x) \cdot b(x) \cdot x^{-\lceil \frac{m}{2} \rceil} \bmod f(x)$

```

1:  $Res(x) = 0, Res_C(x) = 0, Res_M(x) = 0$  ,
2: for  $i$  from 0 to  $\lceil \frac{s}{2} \rceil - 1$  do
3:    $Res_C(x) \leftarrow Res_{C_{n-1}} \cdot f(x) + Res_C(x) \cdot x^w + A_{s-i} \cdot b(x)$ 
4:    $Res_M(x) \leftarrow Res_M(x) + A_i \cdot b(x)$ 
5:    $M(x) \leftarrow Res_{M_0} \cdot F_0'(x) \bmod x^w$ 
6:    $Res_M(x) \leftarrow Res_M(x) + M(x) \cdot f(x)$ 
7:    $Res_M(x) \leftarrow Res_M(x) \div x^w$ 
8: end for
9:  $Res(x) = Res_C(x) + Res_M(x)$ 
10: Return  $Res(x)$ 

```

- Level 2:

1. Normal to Montgomery representation conversion (NtoM)
2. Affine to Projective coordinates conversion (AtoP)
3. EC Point Multiplication (PM)
4. Projective to Affine coordinates conversion (PtoA)
5. Montgomery to Normal representation conversion (MtoN)

- Level 3: EC Point Doubling (PD) and EC Point Addition (PA)

- Level 4: Modular Addition (MA), Modular Multiplication (MM) and Modular Inversion (MI)

The main control Finite State Machine (FSM) first commands the NtoM to convert all the inputs from normal to Montgomery representation and the AtoP to convert the coordinates from affine to projective. The NtoM and the AtoP both use the MM to perform these conversions. When all conversions are finished the Main Controller orchestrates the PM block to start the point multiplication by invoking the PD and the PA in parallel. It writes the resulting X1 and Z1 to its output registers. The FSM inside the PM orchestrates these operations. Due to the parallelism of the point operations both the PD and the PA use an MA and an MM. The next step after point multiplication is conversion from projective to affine representation using the MI, which invokes the MM to perform the modular inversion using Fermat's little theorem [15]. Finally, the affine coordinates are converted from Montgomery to normal representation using the MM.

Figure 1. Architecture of the elliptic curve processor.

5 Results

The results of our design on a Xilinx Virtex XCV800 FPGA are given in Table 1.

Table 1. Results for $n = 179$ where bit-serial and digit-serial multipliers are compared

Results for ECC point multiplication			
	# slices	period (ns)	latency (ms)
$w = 1$	10626	19.165	2.479
$w = 4$	11433	19.298	1.886
$w = 8$	11622	20.050	1.008
$w = 16$	11881	20.961	0.557

Table 2 presents a broader comparison with other architectures. We have included only those FPGA solutions that are either scalable [9, 10] or that are believed to be the state of the art in ECC hardware implementations. We give this comparison as a proof that a scalable and side-channel secure design can also lead to a solution that is competitive in performance.

6 Simple Side-Channel Resistance

Implementations of cryptographic algorithms should be resistant to side-channel attacks such as

Table 2. Comparison with other rel. work for point multiplication in $GF(2^n)$ (here $w=16$)

Ref.	Field	Freq. (MHz)	Lat. (ms)	Hw compl.
[9]	$\mathbb{F}_{2^{160}}$	50.0	5	n. a.
[23]	$\mathbb{F}_{2^{167}}$	76.7	0.84	6 513 g., 501 reg.
[10]	$\mathbb{F}_{2^{160}}$	66.4	0.59	14 241 LUTs, 2 990 FFs
[12]	$\mathbb{F}_{2^{163}}$	22.1	7.4	not known
ours	$\mathbb{F}_{2^{179}}$	47.7	0.557	11 881 sl.

timing [16], power [17] and electromagnetic radiation [25, 8] analysis attacks. These attacks present a realistic threat for wireless applications and have been demonstrated to be very effective against smart cards without specific countermeasures. Here we discuss the ability of our implementation to withstand simple side-channel attacks, such as the Simple Power Analysis (SPA). In that case the attacker can get some information about the secret key by observing one or a few power consumption graphs. To prevent that, cryptographic algorithms should be implemented as sequences of operations that are indistinguishable through simple side-channel analysis. Chevallier-Mames *et al.* define this property as the side-channel atomicity [7]. According to the authors, the SPA-resistant algorithms should consist of so-called side-channel atomic blocks, which are algorithm specific. In our implementation, there exist side-channel atomic blocks on different level of ECC hierarchy. Following the top-down approach those are point addition/doubling and multiplication/squaring. Figure 2 shows a pattern for point addition implemented as in most of the standard references on ECC [11, 4]. In this case the addition takes 14 multiplications, which is visible from the power trace. (The standard doubling takes 10 multiplications in total.) The same situation in the light of Algorithm 2 results in two not so distinguishable patterns (Figure 3 and 4). In both figures the total of 6 field multiplications can be observed.

7 Conclusions

In this paper a complete ECC processor for binary fields is presented. An FPGA implementation has been described. We proposed a fully balanced

Figure 2. Power consumption trace of an EC point addition as in the IEEE standard. The total number of steps is 14 as visible from the graph.

Figure 3. Power consumption trace of an EC point addition as in Algorithm 2 (6 field multiplications are easily detected).

Figure 4. Power consumption trace of an EC point doubling as in Algorithm 2 includes also 6 multiplications.

point multiplication algorithm that performs the same operations for every loop of the algorithm. Furthermore, a new algorithm for field multiplication is given, which performs two separate multiplications in parallel. By using this approach we believe that

this so-called side-channel aware design is the first step towards side-channel resistance. However, it is clear that additional countermeasures will be required to prevent more advanced attacks.

References

- [1] L. Batina, N. Mentens, S. B. Örs, and B. Preneel. Serial multiplier architectures over $\text{GF}(2^n)$ for elliptic curve cryptosystems. In *Proceedings of The 12th IEEE MELECON 2004*, Dubrovnik, Croatia, May 12–15, 4 pages.
- [2] L. Batina, S. B. Örs, B. Preneel, and J. Vandewalle. Hardware architectures for public key cryptography. *Elsevier Science Integration the VLSI Journal*, 34(1–2):1–64, 2003.
- [3] T. Beth and D. Gollmann. Algorithm engineering for public key algorithm. *IEEE Journal on Selected Areas in Communications*, 7(4):458–465, May 1989.
- [4] I. Blake, G. Seroussi, and N. P. Smart. *Elliptic Curves in Cryptography*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.
- [5] E. Brier and M. Joye. Weierstrass Ell. Curves and Side-Channel Attacks. In D. Naccache and P. Pailier, editors, *Proc. of PKC'02*, volume 2274 of *LNCS*, pages 335–345. Springer-Verlag, 2002.
- [6] Ç. K. Koç and T. Acar. Montgomery multiplication in $\text{GF}(2^k)$. *Designs, Codes and Cryptography*, 14:57–69, 1998.
- [7] B. Chevallier-Mames, M. Ciet, and M. Joye. Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. *IEEE Transactions on Computers*, 53(6):760–768, 2004.
- [8] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2162 in Lecture Notes in Computer Science, pages 255–265. Springer-Verlag, 2001.
- [9] 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.
- [10] N. Gura, S. C. Shantz, H. Eberle, D. Finchelstein, S. Gupta, V. Gupta, and D. Stebila. An end-to-end systems approach to elliptic curve cryptography. In Burt Kaliski Jr., Ç. K. Koç, and C. Paar, editors, *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, Lecture Notes in Computer Science 2523, 2002.
- [11] IEEE P1363. Standard specifications for public key cryptography, 1999.
- [12] P. Kitsos, G. Theodoridos, and O. Koufopavlou. An efficient reconfigurable multiplier architecture for Galois Field $\text{GF}(2^m)$. *Elsevier Science Microelectronics Journal*, 34:975–980, 2003.
- [13] D. E. Knuth. *The Art of Computer Programming*, volume 2/Seminumerical Algorithms. Addison-Wesley, 1997.
- [14] N. Koblitz. Elliptic curve cryptosystem. *Math. Comp.*, 48:203–209, 1987.
- [15] N. Koblitz. *A Course in Number Theory and Cryptography*, volume 114 of *Graduate text in mathematics*. Springer-Verlag, Berlin, Germany, second edition, 1994.
- [16] P. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In N. Koblitz, editor, *Advances in Cryptology: Proceedings of CRYPTO'96*, number 1109 in Lecture Notes in Computer Science, pages 104–113. Springer-Verlag, 1996.
- [17] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology: Proceedings of CRYPTO'99*, number 1666 in Lecture Notes in Computer Science, pages 388–397. Springer-Verlag, 1999.
- [18] J. López and R. Dahab. Fast multiplication on elliptic curves over $\text{GF}(2^m)$. 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.
- [19] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Appl. Cryptog.* CRC Press, 1997.
- [20] A. Menezes and S. Vanstone. Elliptic curve cryptosystems and their implementations. *Journal of Cryptology*, (6):209–224, 1993.
- [21] V. Miller. Uses of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology: Proceedings of CRYPTO'85*, number 218 in Lecture Notes in Computer Science, pages 417–426. Springer-Verlag, 1985.
- [22] P. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, Vol. 48:243–264, 1987.
- [23] G. Orlando and C. Paar. A high-performance reconfigurable elliptic curve processor for $\text{GF}(2^m)$. In Ç. K. Koç and C. Paar, editors, *Proceedings of 2nd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 1965 in Lecture Notes in Computer Science, pages 41–56. Springer-Verlag, 2000.
- [24] M. J. Potgieter. A hardware implementation of the group operations necessary for implementing an elliptic curve cryptosystem over a characteristic two finite field. Final report of project EPR400, Technical University Eindhoven, 2002.
- [25] J.-J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In I. Attali and T. P. Jensen, editors, *Smart Card Programming and Security (E-smart 2001)*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer-Verlag, 2001.
- [26] M. Stam. *Speeding up Subgroup Cryptosystems*. PhD thesis, Tech. Universiteit Eindhoven, 2003.