Published in IET Information Security
Received on 25th August 2008
Revised on 16th July 2009
doi: 10.1049/iet-ifs.2009.0038


ISSN 1751-8709

# Efficient and side-channel-aware implementations of elliptic curve cryptosystems over prime fields

*D. Karakoyunlu*[1] *F.K. Gurkaynak*[2] *B. Sunar*[1] *Y. Leblebici*[2]

[1]*Worcester Polytechnic Institute (WPI), CRIS Lab, USA*
[2]*Swiss Federal Institute of Technology (EPFL), LSM, Switzerland*
*E-mail: deniz@wpi.edu*

**Abstract:** Elliptic curve cryptosystems (ECCs) are utilised as an alternative to traditional public-key cryptosystems, and are more suitable for resource-limited environments because of smaller parameter size. In this study, the authors carry out a thorough investigation of side-channel attack aware ECC implementations over finite fields of prime characteristic including the recently introduced Edwards formulation of elliptic curves. The Edwards formulation of elliptic curves is promising in performance with built-in resiliency against simple side-channel attacks. To our knowledge the authors present the first hardware implementation for the Edwards formulation of elliptic curves. The authors also propose a technique to apply non-adjacent form (NAF) scalar multiplication algorithm with side-channel security using the Edwards formulation. In addition, the authors implement Joye's highly regular add-always scalar multiplication algorithm both with the Weierstrass and Edwards formulation of elliptic curves. Our results show that the Edwards formulation allows increased area–time performance with projective coordinates. However, the Weierstrass formulation with affine coordinates results in the simplest architecture, and therefore has the best area–time performance as long as an efficient modular divider is available.

## 1 Introduction

In the mid-1980s Koblitz [1] and Miller [2] independently proposed using elliptic curves for public-key cryptosystems. Since then, elliptic curve cryptosystem (ECC) has been intensively studied, and became popular among other common public-key cryptosystems such as RSA, Diffie–Hellman and ElGamal. In [3], Lenstra and Verheul reported that ECC using a 130-bit key offers comparable security as RSA with a key length of 1024 bits. The shorter parameter size makes ECC especially attractive for embedded applications. However, such devices are more prone to side-channel attacks, since the attacker can procure, isolate and test such a system without being detected [4–8]. Therefore security against side-channel attacks is considered to be vital for ECC deployed in embedded systems, despite degradation of performance. Several techniques were proposed for efficient and side-channel attack aware hardware implementation of ECC [9–12]. Unfortunately, these techniques use either specialised fields or specifically chosen elliptic curves. On the other hand, more generic side-channel attack aware implementations involve more complicated equations [13], demand more hardware [14], or leave the system vulnerable to other types of attacks [15–17]. Hence, providing a high-performance non-specialised implementation, while retaining a degree of side-channel resiliency remains a challenge.

In 2007, Edwards proposed a novel formulation of elliptic curves and associated point arithmetic operations defined over all non-binary fields [18]. Bernstein and Lange analysed and compared the complexity (in number of elementary field operations) of basic group operations for different forms of elliptic curves in various coordinate systems [19]. They suggest that the Edwards elliptic curve formulation has superior performance than the fastest known ECC algorithms. Binary Edwards curves also exist

[20], but they are not in the scope of this paper. In this paper, we present a comprehensive overview and comparison of parameter agnostic hardware implementations of ECC over finite fields of prime characteristics. In particular, we present optimised hardware realisations of ECC in Weierstrass and Edwards formulations using affine and projective coordinates. We compare these implementations in terms of their area and throughput performance. We also realise them in a ECC processor both with CMOS technology, and power balanced MOS current-mode differential logic (MCML) technology [21] that provides differential side-channel attacks (DCA) resiliency.

Furthermore, we introduce techniques for improving the performance at various implementation levels without undermining side-channel awareness. In most ASIC arithmetic units, carry chains cause bottlenecks. Our systematic use of redundant digits for all modular arithmetic operations is a significant advantage for reaching higher operating frequencies, therefore we are setting ASIC speed records for prime-field ECC. To our knowledge, we implement the first hardware implementation of Marc Joye's recently introduced highly regular add-always scalar multiplication algorithm, which is proven to be secure against SPA-type attacks and safe-error attacks [22]. Finally, we introduce a side-channel-aware version of non-adjacent form (NAF) scalar multiplication algorithm for Edwards formulation in Algorithm 1.

The organisation of this paper is as follows. Section 2 provides a preliminary introduction to ECC, defines the main parameters and introduces the new Edwards formulation for ECC. In Section 3, the point multiplication methods with side-channel attack precautions are investigated. The details of the hardware implementation are explained in Section 4. The implementation results are presented in Section 5. Finally Section 6 concludes the paper.

## 2 Elliptic curve cryptosystems

In this section, we will briefly present the ECC formulations over finite fields of prime characteristics. The reader is referred to [23], for a detailed treatment of ECC. In order to construct a cryptographic system, we first need to define a suitable elliptic curve $E$ defined over a prime field $\mathbb{F}_p$ [24]. A cyclic subgroup of $E(\mathbb{F}_p)$ can be generated by selecting a point $P$ of order $n$, and computing its multiples

$$\langle P\rangle = \{\infty,\ P,\ 2P,\ 3P,\ \ldots,\ (\#n-1)P\}$$

The elliptic curve discrete logarithm problem (ECDLP) is defined as determining the value $k \in [1, \#n-1]$, given a point $P \in E(\mathbb{F}_p)$ of order $\#n$, and a point $Q = kP \in \langle P\rangle$. ECDLP is the underlying number theoretical problem used by ECC. In the cryptosystem, a private key is obtained by selecting an integer $k$ randomly from the interval $[1, \#n-1]$. The corresponding public key will be $Q = kP$, and needs to be calculated by scalar-point multiplication.

### 2.1 Simplified Weierstrass formulation for elliptic curves

An elliptic curve $E$ defined over a prime field $\mathbb{F}_p$ (with $p > 3$) can be written in the simplified Weierstrass form as

$$E(\mathbb{F}_p): y^2 = x^3 + ax + b \tag{1}$$

where $a, b \in F_p$, and the discriminant of the curve $\Delta = -16(4a^3 + 27b^2) \neq 0$. A point addition operation can be defined for adding two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ in $E(\mathbb{F}_p)$ resulting in a third point $P + Q = (x_3, y_3)$ in $E(\mathbb{F}_p)$ with the point at $\infty$ serving as identity element $(P + \infty = P)$. Assuming that $P \neq \pm Q$, the point $P + Q = (x_3, y_3)$ can be calculated as

$$\begin{aligned} x_3 &= \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \pmod p \\ y_3 &= \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1 \pmod p \end{aligned} \tag{2}$$

For $P = Q$ the operation is called doubling, and the calculation of $2P = (x_3, y_3)$ is slightly different

$$\begin{aligned} x_3 &= \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \pmod p \\ y_3 &= \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1 \pmod p \end{aligned} \tag{3}$$

Finally, if $P = -Q$ the operation results in point at infinity, and it should be handled separately.

### 2.2 Edwards formulation for elliptic curves

In [18], Edwards showed that elliptic curves over a prime field $\mathbb{F}_p$ (with $p > 3$) in the normal form

$$E(\mathbb{F}_p): x^2 + y^2 = c^2(1 + dx^2y^2) \tag{4}$$

are bi-rationally equivalent to Weierstrass elliptic curves, and can be efficiently transformed from the short Weierstrass form given in (1). The parameter $c$ can be chosen as 1 without loss of generality. Therefore it will be assumed to be 1 in subsequent sections. Bernstein and Lange introduced explicit equations for performing the transformation of the ECC coordinates from Weierstrass to Edwards as well as for performing the group operations on an Edwards curve [19]. The most attractive property of the Edwards formulation is that the same point addition operation can be used even if

the two points on the curve are equal

$$x_3 = \frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2} \pmod{p}$$
$$y_3 = \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2} \pmod{p} \quad (5)$$

Whereas, in the Weierstrass elliptic curve formulation a separate doubling operation as shown in (3) is needed when $P = Q$, and special handling of point at infinity is needed when $P = -Q$. Since only a single type of operation is used, it is reasonable to expect a higher performance from side-channel attack aware ECC implementations using the Edwards formulation when compared to those using the Weierstrass formulation. In addition, in the Edwards formulation, there is no special point at $\infty$, removing another special case that has to be handled by implementations. The Edwards doubling formulation can be further simplified by using the Edwards elliptic curve definition and rewriting $dx^2y^2$ as $x^2 + y^2 - 1$ as suggested by Marc Joye to Bernstein *et al.* in [19]. This optimisation makes the point addition and doubling asymmetric, taking away the side-channel resiliency advantage of unified addition and doubling operations. Nevertheless, Edwards formulation with optimised doubling operations may be utilised with a side-channel aware multiplication algorithm as in the case of Weierstrass formulation

$$x_3 = \frac{2x_1y_1}{x_1^2 + y_1^2} \pmod{p}$$
$$y_3 = \frac{x_1^2 - y_1^2}{x_1^2 + y_1^2 - 2} \pmod{p} \quad (6)$$

## 2.3 Projective coordinate systems

ECC implementations may be viewed at several layers. At the point level the main operation is the scalar-point multiplication, which is realised with multiple point additions and point doubling operations. Each point addition and doubling involves a number of elementary modular arithmetic operations. Modular addition and subtraction are relatively straightforward to implement. Modular multiplication is a reasonably costly operation. At the arithmetic level the implementation of the modular inversion is the most costly operation. The high cost of modular inversion has motivated the investigation of alternative coordinate representations, which avoid the inversion operation at a cost of increased number of field multiplications and additions. The classical formulation where a point $P$ on an elliptic curve $E$ is represented by a pair of elements $(x, y)$ is known as the affine coordinate representation. The affine coordinates can be transformed into projective coordinates that use three elements to represent a point $(X, Y, Z)$, allowing the numerator and the denominator to be calculated separately.

A number of projective coordinate transformations have been proposed in the literature: homogeneous projective, Jacobian, Chudnovsky Jacobian [25], Lopez-Dahab [26] and mixed coordinates [27]. Homogeneous projective coordinates are rarely used in Weierstrass formulation, since the number of multiplications required in exchange of avoiding the inversion is too high. However, Jacobian projective coordinates turn out to be more efficient and most commonly applied either as is, or in a mixed form with affine coordinates. On the other hand, because of the balanced form of equations, homogeneous projective coordinate work well on Edwards elliptic curves.

A Weierstrass elliptic curve defined in (1) is converted to Jacobian coordinates as follows

$$E(\mathbb{F}_p): Y^2 = X^3 + aXZ^4 + bZ^6$$

where $X = xZ^2$, $Y = yZ^3$. Then the point addition (7) and doubling (8) formulations with Jacobian coordinates become [25]

$$\begin{aligned} X_3 &= (Y_2Z_1^3 - Y_1Z_2^3)^2 - (X_2Z_1^2 - X_1Z_2^2)^2 \\ &\quad \times (X_2Z_1^2 + X_1Z_2^2) \pmod{p} \\ 2Y_3 &= (Y_2Z_1^3 - Y_1Z_2^3)[(X_2Z_1^2 - X_1Z_2^2)^2(X_2Z_1^2 + X_1Z_2^2) \\ &\quad - 2X_3] - (X_2Z_1^2 - X_1Z_2^2)^3(Y_2Z_1^3 + Y_1Z_2^3) \pmod{p} \\ Z_3 &= (X_2Z_1^2 - X_1Z_2^2)Z_1Z_2 \pmod{p} \end{aligned} \quad (7)$$

$$\begin{aligned} X_3 &= (3X_1^2 + aZ_1^4)^2 - 8X_1Y_1^2 \pmod{p} \\ Y_3 &= (3X_1^2 + aZ_1^4)(4X_1Y_1^2 - X_3) - 8Y_1^4 \pmod{p} \\ Z_3 &= 2Y_1Z_1 \pmod{p} \end{aligned} \quad (8)$$

The addition formulation can be optimised by removing $Z_2$ values, if one of the points is affine (i.e. $Z_2 = 1$), resulting in the so-called mixed point addition. An Edwards elliptic curve defined in (4) is converted to homogeneous projective coordinates as follows

$$E(\mathbb{F}_p): X^2 + Y^2 = Z^4 + dX^2Y^2$$

where $X = xZ$, $Y = yZ$. The following formulas compute the unified point addition and doubling (9), and optimised doubling (10) operations with projective coordinates [19]. Similar to Jacobian coordinates, addition can be optimised in the case of mixed coordinates

$$\begin{aligned} X_3 &= Z_1Z_2(X_1Y_2 + Y_1X_2)(Z_1^2Z_2^2 - dX_1X_2Y_1Y_2) \pmod{p} \\ Y_3 &= Z_1Z_2(Y_1Y_2 - X_1X_2)(Z_1^2Z_2^2 + dX_1X_2Y_1Y_2) \pmod{p} \\ Z_3 &= (Z_1^2Z_2^2 - dX_1X_2Y_1Y_2)(Z_1^2Z_2^2 + dX_1X_2Y_1Y_2) \pmod{p} \end{aligned} \quad (9)$$

$$\begin{aligned} X_3 &= 2X_1Y_1(X_1^2 + Y_1^2 - 2Z_1^2) \pmod{p} \\ Y_3 &= (X_1^2 - Y_1^2)(X_1^2 + Y_1^2) \pmod{p} \\ Z_3 &= (X_1^2 + Y_1^2)(X_1^2 + Y_1^2 - 2Z_1^2) \pmod{p} \end{aligned} \quad (10)$$



*IET Inf. Secur.*, 2010, Vol. 4, Iss. 1, pp. 30–43
doi: 10.1049/iet-ifs.2009.0038

## 3 Side-channel attack aware point multiplication

The so-called binary multiplication methods provide a systematic way of ordering the addition and doubling operations. In a typical binary multiplication scheme, as the bits of the multiplicand are processed sequentially, a point doubling is performed for each bit, and a point addition is performed if the current bit is equal to one. Hence, the run time of the binary multiplication scheme depends on the number of non-zero bits of the multiplicand. On average, for a multiplicand of bit length $n$, the point multiplication requires $n$ doubling operations and $n/2$ point additions. A more advanced binary multiplication method requires the scalar multiplicand to be recorded into the non-adjacent signed digit form (NAF) [28]. A NAF binary number will not have two consecutive non-zero digits (1 or $-1$ in signed digit form), reducing the number of point additions to less than $n/2$ throughout an $n$-bit scalar-point multiplication ($n/3$ point additions on average). Although the number of doublings remains constant, the NAF method leads to a modest linear improvement in the number of point additions.

Both the standard binary multiplication scheme and the NAF scheme conditionally perform a point addition (or subtraction) driven by the binary digit values of the secret multiplicand. Side-channel characteristics of distinguishable point addition and doubling operations can be observed to vary while a point multiplication is carried out, which can potentially be used by an adversary to reveal parts of the secret key [29, 30]. Moreover, even if the point addition and doubling operations are indistinguishable as in unified Edwards formulation, the total run time of the point multiplication will still depend on the number of non-zero binary digits of the multiplicand, since point addition is only carried out when a digit is non-zero. In this case, an attacker observing the run time, could determine the Hamming weight of the multiplicand, reducing the possible solution space significantly. Although it seems impossible to foresee all possible side-channel attacks that might emerge in the future, we believe that cryptographic architectures should be designed to withstand the attacks summarised above. We call such architectures side-channel aware.

Side-channel attacks can be classified into simple side-channel attacks (SCA), which directly interpret data characteristics that are visible in a single or a few measurement traces, and differential side-channel attacks (DCA), which interpret the side-channel differences of correlated measurements. DCA can be further enhanced by applying statistical methods over a template of measurement traces [31].

### 3.1 SCA countermeasures for distinguishable point operations

When the point addition and doubling operations are different, the only way to make a point multiplication side-channel attack aware is using a uniform sequence of point operations that do not depend on the value of the multiplicand. A method proposed by Moller performs point multiplication with fixed pattern of doublings and additions with less than $2n$ point operations in total [17], but it involves a fixed look-up table that makes the system susceptible to statistical attacks described in [32]. Recognising this problem, he proposes a new method to avoid a fixed table [14], employing a randomised initialisation stage to achieve resistance against side-channel attacks. However, when a random number generator is not incorporated, reducing the number of point operations is not possible. One has to use a regular multiplication algorithm that involves one point addition and one point doubling for each binary digit of the multiplicand to avoid revealing the order and number of the non-zero digits.

One solution to achieve a regular multiplication algorithm is to introduce point addition operations when the binary digit is zero [15, 16], or inserting dummy atomic operations to achieve side-channel atomicity [33]. However, this is not always a trivial task. If the operations are dummy, they are vulnerable to fault insertion attacks, where the attacker deliberately introduces a fault during one operation and monitors the output for a change. If the correct output is produced in presence of faults, the attacker will be able to conclude that the operation where the fault was introduced was a dummy operation [34, 35].

The so-called Montgomery binary ladder [36] protects against SCA and fault insertion attacks, since it is highly regular and does not involve dummy operations. Recent studies have shown that processing the bits of multiplicand from left-to-right, as Montgomery ladder does, are also vulnerable to certain attacks [37, 38]. In 2007, Joye introduced the add-always (also referred as: always add-and-double algorithm) binary scalar multiplication algorithm [22]. This new algorithm is highly regular, processed from right-to-left, and it requires no precomputation or prior recoding. Add-always multiplication algorithm requires $n$ point doublings and $n$ point additions regardless of the value of the scalar multiplicand, and two temporary registers are needed to store the results of each iteration. It should be noted that the standard left-to-right algorithm with dummy operations allows us to accumulate the multiplication result in only one register, and use mixed coordinates since the coordinates of the input point is kept intact ($Z$-coordinate of the input point will be 1). Nevertheless, dummy operations and left-to-right processing should be avoided, because of the vulnerabilities described above.

### 3.2 SCA countermeasures for unified point operations

In the case of distinguishable point addition and doubling operations, a side-channel attack aware point multiplication requires using a regular point multiplication algorithm that consists of $n$ doubling operations and $n$ point additions for a

multiplicand of bit length $n$. The Edwards formulation allows unified point addition and doubling without requiring specialised elliptic curves, or any randomisation or initialisation stage. When the point addition and doubling operations are unified, even if the scalar-point multiplication algorithm is irregular it will not cause side-channel leakage as long as the total number of operations is constant. As it is mentioned in the beginning of this section, the NAF point multiplication algorithm always requires fewer than $n/2$ point additions. By carrying out necessary number of extra operations after finishing the point multiplication, the total run-time could be set to the worst case in order to prevent dependency on the multiplicand value. However, if these extra operations do not update the value of the result, the system will be vulnerable to fault insertion attacks as explained above. In Fig. 1, we propose a method that computes the extra operations at the end of a NAF multiplication. Since the additional operations do affect the computed result, the algorithm is also robust against fault insertion attacks. The first seven lines of the algorithm compute the NAF point multiplication, with the result stored in $R_0$. Note that the addition and subtraction operations in lines 4 and 5 are virtually the same operations on elliptic curves. Moreover, our implementation uses radix-2 signed digit redundant representation, which makes it even simpler to achieve indistinguishable addition and subtraction operations. All we need to is to swap the wiring of 2-bit representation of the second operand digits in the case of point subtraction. In line 8, the number of necessary extra operations is calculated and stored in $r$, and the register $R_0$ is updated by the sum of $R_0$ and $R_1$. After this addition, the result can be expressed as: result $= R_0 - R_1$. Throughout the for-loop in lines 9–12, both $R_0$ and $R_1$. are continuously updated for $r/2$ iterations, so that $2\lfloor r/2 \rfloor$ extra operations are carried out, while result $= R_0 - R_1$ still holds. Finally in line 12, the result is recomputed by the subtraction: $R_0 - R_1$. The addition in line 13 is conditionally performed in order to achieve $r + 2$ extra point additions in total regardless of $r$ being odd or even. Hence, the computations will end after $(3n/2) + 2$ unified point operations.

### 3.3 DCA countermeasures

Differential side-channel analysis allows more powerful attacks that succeed even in the presence of SCA countermeasures [39]. Comprehensive information about performing differential side-channel analysis can be found in [40]. Several classes of countermeasures were proposed against DCA, for example using noise generators to confuse attackers by adding random noise to the power signature [41], feeding idle datapath units with random data to provide a more uniform data profile [42], using masking techniques [43, 44], and finally using power balanced IC libraries that have data-independent power consumption characteristics [45–47]. For enhanced robustness against side-channel attacks, the design may be synthesised with such precautions at the circuit level. We synthesised our design both with standard CMOS technology, and with the power balanced MOS current-mode differential logic (MCML) technology [21].

At the algorithm level, DCA resiliency can be achieved by multiplying the point by a random number prior to each point addition or doubling with projective coordinates. In Section 2.3, we have stated that homogeneous projective coordinates ($xZ$, $yZ$, $Z$) are more suitable for the Edwards formulation, whereas the Jacobian coordinates, that is $(xZ^2, yZ^3, Z)$ are more suitable for the Weierstrass formulation. Prior to each point operation randomisation can be carried out by replacing the point coordinates ($X$, $Y$, $Z$) with ($\lambda X$:$\lambda Y$:$\lambda Z$) in the case of homogeneous projective coordinates, and with ($\lambda^2 X$:$\lambda^3 Y$:$\lambda Z$) in the case of the Jacobian coordinates, where $\lambda \neq 0$ is a random number [15]. Hence, if projective randomisation is utilised, the Edwards formulation has further performance benefit

**Algorithm 1** Side-channel attack aware NAF scalar multiplication algorithm

Inputs: $P \in F_p$ and $k = (k_{n-1}, \ldots, k_0)_2 \in N$
Output: $Q = kP \in F_p$

1: $R_0 := 0$; $R_1 := P$; $a := 0$; $r := 0$;
2: **for** $j = 0$ **to** $n - 1$ **do**
3: Recode $k_j$ on the fly into non-adjacent signed-digit form with Reitwiener's method [50].
4: **if** $(k_j = 1)$ **then** $R_0 := R_0 + R_1$; $a := a + 1$; **end if**
5: **if** $(k_j = -1)$ **then** $R_0 := R_0 - R_1$; $a := a + 1$; **end if**
6: $R_1 := R_1 + R_1$;
7: **end for**
8: $R_0 := R_0 + R_1$; $r := \frac{n}{2} - a$;
9: **for** $j = 0$ **to** $\frac{r}{2}$ **do**
10: $R_0 := R_0 + R_1$; $R_1 := R_1 + R_1$;
11: **end for**
12: $R_0 := R_0 - R_1$;
13: **if** $r$ **is** *odd* **then** $R_1 := R_1 + R_1$; **end if**
14: **return** $R_0$

**Figure 1** *Side-channel attack aware NAF scalar multiplication algorithm*



*IET Inf. Secur.*, 2010, Vol. 4, Iss. 1, pp. 30–43
doi: 10.1049/iet-ifs.2009.0038

of requiring fewer field multiplications since there is no need for computing the square and cube of the random number prior to each point addition or doubling. We do not apply projective randomisation, since we do not want to incorporate a random number generator in our hardware implementation.

# 4 Hardware implementation

## 4.1 Modular arithmetic operations

Modular arithmetic operations are the core operations of ECC. In order to improve the performance of the overall system, it is crucial to optimise the modular arithmetic operations by avoiding side-channel information leakage because of data-dependent run time. As the first step of ECC implementation, we have designed the following optimised modular arithmetic components:

*4.1.1 Carry-propagation-free addition with signed digit redundant representation:* Addition is the primary building block in implementing arithmetic operations. If addition is slow or area expensive, all other operations suffer from this. In order to achieve parallel addition of two $n$-digit redundant binary numbers in constant time without carry propagation, we used the $n$-digit radix-2 signed digit (SD2) representation that uses the digit set $\{-1, 0, 1\}$ [48], and carry-propagation-free addition as proposed in [49]. Fig. 2 shows addition of two consecutive SD2 digits using carry-save adders. It can be observed that signals do not propagate through more than two full adders. The $n$-digit redundant binary adders (RBAs) are realised by cascading 4-to-2 signed-digit carry-save adders as presented in [50], which allowed us to keep the critical delay path of computing $n$-digit addition within the delay of two full adders. The RBA is used as the primary building block in the implementation of the modular arithmetic operations.

*4.1.2 Radix-4 Montgomery multiplication algorithm:* A constant run-time radix-2 Montgomery modular multiplier that uses redundant representation is presented in [51]. We have modified this algorithm to perform multiplication in radix-4, reducing the run time by a factor of 2. This multiplier works by computing five steps for each radix-4 digit as presented in Fig. 3, where LSD stands for least significant digit.

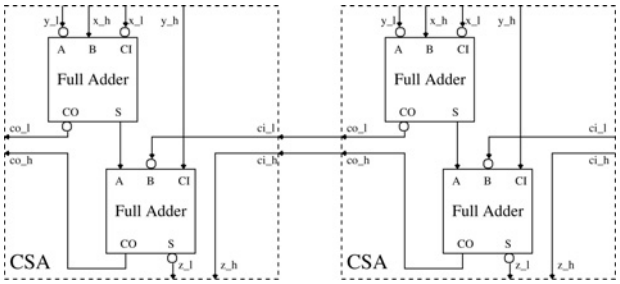


**Figure 2** *Addition of two consecutive SD2 digits using carry-save adders*

In Fig. 3, step 2 requires only a single or double left shift of SD2 digits, whereas steps 3 and 4 require $n$-digit redundant binary addition operations. The modular multiplier completes the $n$-bit multiplication in $(n/2)+1$ iterations through two RBA stages. Obviously, the multiplication result will be in Montgomery residue form (divided by $2^{n+2}$). A side effect of using a radix-4 multiplier is that the range of operand values has increased from $(-p, p)$ of original algorithm in [57] to $(-2p, 2p)$, where $p$ is the prime modulus.

*4.1.3 Extended binary GCD modular division algorithm:* Modular division is the most costly operation in ECC operations, which is usually avoided by using projective coordinates to trade several additional multiplications with division at every point addition cycle. If an efficient division unit could be implemented, the additional complexity incurred because of projective coordinates can be avoided. In order to achieve a high-performance divider, we implemented the modular division presented in Fig. 4. This algorithm computes the GCD of the divisor and the prime modulus, which is equal to 1. Meanwhile, the same operations are applied to the dividend in parallel with a modulus reduction after each iteration. When the algorithm terminates by computing the GCD of the divisor and prime modulus as 1, the same operations applied to the dividend effectively computes the

**Algorithm 2** Radix-4 montgomery multiplication algorithm

Inputs: A:= Multiplier, B:=Multiplicand, M:= Modulus
Output: R:= Result

```
R:= 0;
for i from 1 to n/2
  Step 1: a:= LSD(A), A≫2;
  Step 2: P:= a*B; (where: a ∈ {−2, −1, 0, 1, 2})
  Step 3: R:= R+P;
  Step 4: R:= R+q*M; (where: q ∈ {−2, −1, 0, 1, 2}, so that LSD(R) = 0)
  Step 5: R≫2;
end for
return R;
```

**Figure 3** *Radix-4 Montgomery multiplication algorithm*

**Algorithm 3** Extended binary GCD modular division algorithm [26]

Inputs: A:= Dividend, B:=Divisor, M:= Modulus
Output: Q:= Quotient

```
p:= n;   d:= 0;   Q:= 0;
while p≠ 0 do
  while B is even do
     B:=B/2;   A:=A/2 mod M;
     p:=p−1;   d:=d−1;
  end while
  if d< 0 then
     swap(B,M);   swap(A,Q);   d := −d;
  end if
  B:=(B+k*M)/4;   A:=(A+k*Q)/4 mod M; (where: k ∈{−1,1}, so that (B+k*M) mod 4= 0)
  p:=p−1;   d:=d−1;
end while
if M= −1 then Q:=M−Q; end if
return Q;
```

**Figure 4** *Extended binary GCD modular division algorithm*

quotient of the modular division. The binary GCD algorithm is further optimised by observing the facts that the prime modulus is always an odd number; and when both numbers are odd, either their sum or their difference is a multiple of 4. Hence it reduces to following two cases, where B is the divisor and M is the prime modulus:

```
If B is even, M is odd: GCD(B,M):=GCD(B/2,M)

If B is odd, M is odd: GCD(B,M):=GCD([B+M]/4,M)
```

The modular divider completes $n$-bit division in only $2n + 4$ iterations in the $(-2p, 2p)$ range. Constant run time for side-channel awareness is achieved by continuing the iterations until the control register reaches the end as suggested in [51]. For each iteration an adder is required for computing the GCD of the divisor and the prime modulus, and a modular adder is required for applying the same operation to the dividend in parallel together with modulus reduction. Therefore a total of three RBA stages are necessary (one RBA for regular addition and two RBAs for modular addition). It should also be noted that the arithmetic operations are carried out in the Montgomery residue format, and division does not preserve the Montgomery residue form of the operands. Therefore a division should be followed by a multiplication to transform the result back into Montgomery residue form, which increases the effective division time to $(5n/2) + 6$ clock cycles.

*4.1.4 Modular addition and subtraction algorithm:* The modular addition method described in [52] allows computing the $n$-bit modular addition or subtraction via a regular addition followed by a modular correction step that depends on checking only the most significant three digits of the intermediate result. Hence, modular addition and subtraction can be computed in a single iteration through two RBA stages. This method was also modified to work for $(-2p, 2p)$ range instead of $(-p, p)$ range, in order to achieve consistency with the multiplication and division.

*4.1.5 Combined modular arithmetic units:* The modular arithmetic units described above are all based on RBAs, and therefore efficient resource sharing is possible. We have implemented two different arithmetic units. The first arithmetic unit (mmu) is capable of modular multiplication, addition and subtraction; and it is intended to be used in projective point operations. The second unit (mau) is additionally capable of modular division; and it is intended to be used in affine point operations. The first arithmetic unit (mmu) requires two RBA stages, whereas the second unit (mau) requires three RBA stages in order to accommodate for division.

When the point multiplication operation is carried out with projective coordinates, a final division is necessary to have the resulting point in affine coordinates. We realised this operation by taking the modular inverse of the $Z$-coordinate using multiplications according to Fermat's theorem: $Z^{-1} = Z^{p-2} \pmod p$, if $\gcd(Z,p) = 1$ [53]. Although this inversion takes much longer $(3n^2/4) + 3n$ on average) than the extended binary GCD division algorithm, it is carried out only once at the end of a point multiplication. Therefore the performance gain in terms of area is more than the performance loss in terms of time, and it turns out to be more area–time efficient. The number of clock cycles for different modular operations and the hardware resources in each modular arithmetic unit is shown in Table 1.

## 4.2 Point addition and doubling operations

We have designed four different point addition and doubling units (Weierstrass affine, Weierstrass projective, Edwards




doi: 10.1049/iet-ifs.2009.0038

**Table 1** Number of clock cycles for the operations of combined modular arithmetic units

| Unit | Multiplication | Addition and subtraction | Division | Resources |
|---|---|---|---|---|
| mmu | $\frac{n}{2}+2$ | 1 | $\frac{3n^2}{4}+3n$ | 2 RBAs |
| mau | $\frac{n}{2}+2$ | 1 | $\frac{5n}{2}+6$ | 3 RBAs |

affine, Edwards projective), which implement the addition and doubling formulations in (2), (3) and (5)–(10). In addition to point operations, each unit is also capable of necessary initial and final transformations. The initial transform computes the Montgomery residue forms of the point coordinates. The final transform computes the inverse Montgomery transformation and projective-to-affine transformation for projective coordinates. We also take advantage of the homogeneity in Edwards projective operations by avoiding Montgomery transformations. This is possible since the Montgomery modular multiplications in non-residue form do not affect the $X/Z$ and $Y/Z$ ratios at the end of a point addition or doubling.

For Weierstrass affine unit, we cannot utilise more than one arithmetic unit, because of the data dependence in point operations. For the others, having multiple units in parallel is possible. The single unit case always has the best area–time product, since all parallel units may not be utilised with the same type of operation in all stages of the dataflow. Having a single arithmetic unit is also preferable because of the limited area resources of ECC applications. The Weierstrass implementations require special handling of point at infinity. During each Weierstrass point operation, a check for whether the resulting point will be point at infinity is performed as well. This check is carried out offline (off the critical path delay) through a $(n/4)$-bit comparator without stalling the point operation. Selecting the comparator size as $(n/4)$-bit also allows keeping the area overhead low. For bit values less than 256, the comparator will require three-levels of $(1+4+16=21)$ four-input XOR gates.

Each unit is implemented with a datapath that consists of an appropriate modular arithmetic unit, a set of input selection multiplexers and temporary registers, and a control unit. The control units for each point operation are designed with finite state machine strategy, where each state corresponds to an arithmetic operation through register-to-register dataflow. For each point operation, the arithmetic operations are scheduled to require minimum number of temporary storage registers after a careful data dependence analysis. A generic point addition and doubling unit is shown in Fig. 5. Table 2 shows the hardware resources, and the cost of point operations in terms of modular arithmetic operations for each unit. Looking at the operation counts, we observe that operations with projective coordinates have better performance in the Edwards formulation, whereas operations with affine coordinates have better performance in the Weierstrass formulation. Meanwhile, the comparison between Weierstrass affine and Edwards projective operations depend on the performance ratio of division and multiplication. In terms of area, the projective units have the advantage of using smaller arithmetic units. However, they need more registers because of the increased complexity of the point operations with projective coordinates, and additional storage requirement for $Z$-coordinates. Therefore projective coordinates may not necessarily provide an area advantage. The overall gate count percentages of the arithmetic units for each ECC point multiplication are provided in Table 4 in Section 5. Since the number of flip-flops in state machine registers are much less than storage registers, the area percentage of control units are negligible in size. Therefore the remaining area percentage is allocated mostly for the storage registers.

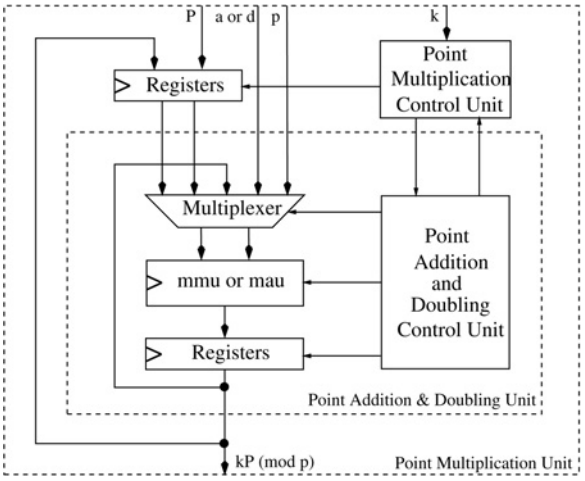


**Figure 5** *Block diagram of ECC point multiplication unit*

**Table 2** Number of modular arithmetic operations for point addition and doubling units

| Unit | Weierstrass affine | Weierstrass Jacobian | Edwards affine | Edwards projective |
|---|---|---|---|---|
| resources | 1 mau, 1 reg | 1 mmu, 5 regs | 1 mau, 3 regs | 1 mmu, 4 regs |
| point addition | 1CD + 2M + 7A | 16M + 9A | 2CD + 5M + 7A | 12M + 7A |
| mixed point addition | not applicable | 11M + 9A | not applicable | 11M + 7A |
| point doubling | 1CD + 3M + 9A | 10M + 13A | 2CD + 3M + 4A | 7M + 5A |
| initial transform | 2M | 2M + 2A | 2M | no operation |
| final transform | 2M | 1ED + 4M | 2M | 1ED + 2M |

CD: cheap division using mau; ED: expensive division using mmu; M: multiplication; A: addition and subtraction

**Table 3** Number of arithmetic operations required for point multiplication in different ECC configurations

| Algorithm: | Add-always | Secure NAF |
|---|---|---|
| number of operations: | $n$ additions, $n$ doublings | $3n/2$ unified additions |
| Weierstrass affine | $(2n)$CD + $(5n+4)$M + $(16n)$A | not safe |
| Weierstrass Jacobian | (1)ED + $(26n+6)$M + $(22n+2)$A | not safe |
| Edwards affine with unified doublings | $(4n)$CD + $(10n+4)$M + $(14n)$A | $(3n)\text{CD} + \left(\frac{15n}{2}+4\right)\text{M} + \left(\frac{21n}{2}\right)\text{A}$ |
| Edwards affine with optimised doublings | $(4n)$CD + $(8n+4)$M + $(11n)$A | not Safe |
| Edwards projective with unified doublings | (1)ED + $(24n+2)$M + $(14n)$A | $(1)\ \text{ED} + (18n+2)\text{M} + \left(\frac{21n}{2}\right)\text{A}$ |
| Edwards projective with optimised doublings | (1)ED +$(19n+2)$M + (12n)A | not Safe |

CD: cheap division using mau; ED: expensive division using mmu; M: multiplication; A: addition and subtraction

### 4.3 Point multiplication operations

To have a complete design space, we have implemented the add-always algorithm for all different ECC configurations. In addition, the secure NAF algorithm presented in Fig. 1 was implemented for the Edwards formulation with unified addition and doubling operations. Table 3 shows the run times of the multiplication algorithms with the modular arithmetic operation costs of point additions and doublings taken from Table 2. Fig. 5 shows the generic block diagram of a point multiplication unit. Each individual system consists of optimised datapath and control units to implement the point multiplication operations.

## 5 Implementation results and performance comparison

Most ECC hardware implementations in the literature have been realised over binary fields. There are only a small number of hardware implementations targeting prime fields mostly implemented in FPGA. We found two comparable ASIC implementations that were recently published. Both implementations were realised using 0.13 μm CMOS technology for bit length of $n = 192$ with no wireload model included. Moreover, they do not address side-channel security. To be able to make fair comparison with our work, we synthesised our design for bit length of $n = 192$. Our implementations were synthesised using UMC 0.18 μm CMOS technology with a target clock of 3 ns and with no wireload model.

Table 4 and Fig. 6 show that the Weierstrass affine system has the best area–time performance among our implementations, whereas the Weierstrass projective system has the worst area–time performance. In comparison to other implementations, seven out of our eight implementations have less area, and four out of our eight implementations have better timing, although we use a slower technology, and address side-channel security as well.

To be able to make a performance comparison among our eight individual implementations as well as with other ASIC implementations, we used post-synthesis results.




doi: 10.1049/iet-ifs.2009.0038

**Table 4** Comparison of synthesis results for ECC point multiplication [GF(p) 192-bit]

| ECC system | Multiplication algorithm | Platform | Max frequency, MHz | Operation time, ms | Area | Area × operation time |
|---|---|---|---|---|---|---|
| Weierstrass affine | add-always | 0.18 μm CMOS | 333.3 | 0.86 | 91K gates (mau: 55.0%) | 78.26 gates × s |
| Weierstrass projective | add-always | 0.18 μm CMOS | 333.3 | 1.59 | 110K gates (mmu: 28.4%) | 174.90 gates × s |
| Edwards affine | add-always with unif. doublings | 0.18 μm CMOS | 333.3 | 1.48 | 94K gates (mau: 53.9%) | 139.12 gates × s |
| Edwards affine | add-always with opt. doublings | 0.18 μm CMOS | 333.3 | 1.42 | 94K gates (mau: 53.9%) | 133.48 gates × s |
| Edwards affine | secure NAF | 0.18 μm CMOS | 333.3 | 1.12 | 93K gates (mau: 54.5%) | 104.16 gates × s |
| Edwards projective | add-always with unif. doublings | 0.18 μm CMOS | 333.3 | 1.47 | 92K gates (mmu: 34.3%) | 135.24 gates × s |
| Edwards projective | add-always with opt. doublings | 0.18 μm CMOS | 333.3 | 1.18 | 92K gates (mmu: 34.3%) | 108.56 gates × s |
| Edwards projective | secure NAF | 0.18 μm CMOS | 333.3 | 1.13 | 92K gates (mmu: 34.3%) | 103.96 gates × s |
| Satoh *et al.* [54] | NAF | 0.13 μm CMOS | 137.7 | 1.44 | 110K gates | 158.40 gates × s |
| Sozzani *et al.* [55] | add and double | 0.13 μm CMOS | 294 | 1.33 | 108K gates | 143.64 gates × s |

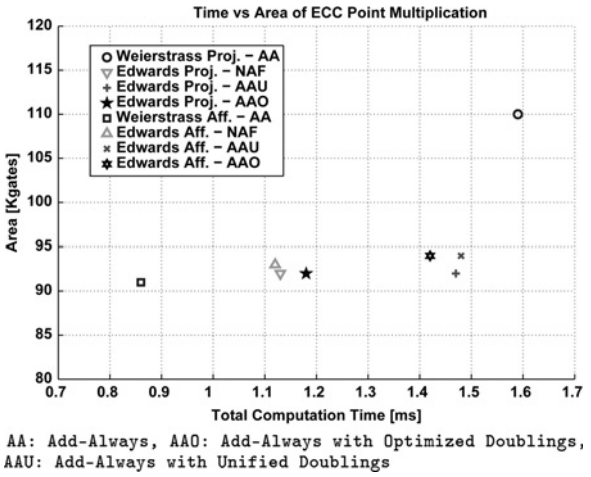


**Figure 6** *Time-area space of ECC point multiplication*

However, these results are not realistic for post-placement and routing, and they do not include the overhead of I/O interface of an ASIC chip. We combined all our implementations as an ECC processor that is capable of all different versions of point multiplication algorithms, and realised it on a single chip. The ECC processor includes a single datapath, control units for four different types of point addition and doubling operations (Weierstrass affine, Weierstrass projective, Edwards affine, Edwards projective), and control units for two different types of point multiplication operations (add-always, secure NAF). Moreover, shift registers were added in top-level in order to interface I/O ports of the processor with the limited number of pins on the chip. The placed and routed chip, realised with UMC 0.18 μm CMOS technology, has a critical path delay of 6.8 ns and cell area of 1.29 $mm^2$, and the one realised with 0.18 μm MOS MCML technology [21] has a critical path delay of 15 ns and cell area of 6.99 $mm^2$. MCML technology has a power balanced IC library, and provides DCA resiliency as mentioned in Section 3.3. Finally, we are also presenting the P&R results of our two best implementations for bit lengths of $n = 160$ and 192, together with several FPGA implementation results in Table 5.

**Table 5** P&R results for ECC point multiplication

| ECC system | GF(p) Field | Platform | Max frequency, MHz | Operation time, ms | Area |
|---|---|---|---|---|---|
| Weierstrass Affine | 160 | 0.18 μm CMOS | 147 | 1.37 | 86K gates |
| Edwards Projective | 160 | 0.18 μm CMOS | 147 | 1.80 | 86K gates |
| Weierstrass Affine | 192 | 0.18 μm CMOS | 147 | 1.95 | 100K gates |
| Edwards Projective | 192 | 0.18 μm CMOS | 147 | 2.56 | 101K gates |
| Ors *et al.* [56] | 160 | XCV1000E-8 | 91.3 | 14.4 | 115.5K gates |
| Mentens *et al.* [57] | 160 | XC3S5000-5 | 66 | 26.8 | 4826 slices, 66 RAMs, 66 mults |
| Mcivor *et al.* [58] | 256 | XC2VP125-7 | 34.46 | 3.86 | 15 755 slices, 256 mults |
| Sakiyama *et al.* [59] | 256 | XC3S5000-5 | 40 | 17.7 | 27 597 slices |

## 6 Conclusions

In this paper, we presented a comprehensive investigation of side-channel attack aware ECC implementations over finite fields of prime characteristics. Our comparison also includes the recently introduced Edwards elliptic curves. Many optimisations have been previously proposed for the Weierstrass formulation, aimed at reducing the cost of point addition and doubling operations, as well as decreasing the number of point additions and doublings. Unfortunately, most of these optimisations make the system prone to side-channel attacks. Optimisations trying to reduce the number of point additions cause the system to leak information about the secret-key and should be avoided. Use of mixed coordinates, which reduces the cost of point addition with projective coordinates is also not possible in a secure multiplication. Consequently, it is not possible to reduce the cost of point additions or the number of point additions in a side-channel attack aware ECC with the generic Weierstrass formulation. With the introduction of Edwards formulation for elliptic curves, on the other hand, it is possible to reduce the number of point additions by using a secure version of NAF. Moreover, Edwards formulation allows simpler operations for point addition and doubling in projective coordinates.

We have shown that the recently introduced Edwards elliptic curve formulation has a number of advantages both for performance and side-channel security. However, the performance benefits are only applicable for projective coordinates, since the affine Edwards formulation is more complicated than Weierstrass formulation. Therefore if an efficient divider is available, Weierstrass affine formulation still offers the best performance. Using projective coordinates has both positive and negative effects on the total cell area. Avoiding the implementation of a divider, which is generally costly, allows reducing the area. However, more complicated point addition and doubling operations require using more temporary registers. Moreover, using three instead of two coordinates to represent a point, increases the area to store the temporary points during point multiplication. We have also shown that side-channel attack aware point multiplication algorithms require holding two temporary points. The adverse effects of all these additional register requirements are exacerbated when each binary digit is stored in 2 bits because of the redundant binary representation to implement fast modular operations. Finally, with the use of efficient resource sharing between divider and multiplier, the area cost of implementing the divider is reduced. Indeed, we have seen that the area cost of additional register usage cancelled out the area gain of not implementing a divider. Thus, the area of affine and projective systems have almost same values that are in the range 91K–94K, except the Weierstrass Projective implementation. Weierstrass Projective system has the most complicated formulation, and it consumes an area of 110K even without a divider. Having lost the advantage of area reduction, the only attraction to use projective coordinates would be timing optimisation by avoiding divisions. However, timing optimisation with projective coordinates is only possible if division operation could be traded for enough number of multiplications. In our implementation, the division/multiplication timing ratio is only 4 and not enough to make the projective operations significantly faster. Therefore we obtain the time costs presented in Table 4 and Fig. 6.

The recently introduced Edwards formulation is shown to be faster than previous elliptic curve formulations in software




doi: 10.1049/iet-ifs.2009.0038

[19]. We show that, in an efficient hardware implementation Weierstrass formulation with affine coordinates offer the best performance because of its simplicity, and Edwards supersedes Weierstrass formulation only when projective coordinates are used. The fact that affine systems offer better performance than projective systems was also pointed out in [60–62], which demonstrate that it is profitable to investigate inversion architectures since the affine coordinates provide superior performance when the inversion operation is realised in hardware.

## 7 Acknowledgments

The authors would like to express their thanks Dr. Marcelo Kaihara and Prof. Arjen Lenstra for their support, assistance and inspiration during the implementation period of this work. Moreover, special thanks to IET Information Security Editorial Office and anonymous reviewers for their contributions to improve the manuscript. This work was partially supported by National Science Foundation through NSF Cybertrust Grant No. 0831416 and NSF-ANI-Career Grant No. 0133297.

## 8 References

[1] KOBLITZ N.: 'Elliptic curve cryptosystems', *Math. Comput.*, 1987, **48**, (177), pp. 203–209

[2] MILLER V.S.: 'Use of elliptic curves in cryptography'. Advances in Cryptology (CRYPTO 1985), Santa Barbara, CA, USA, 1985, vol. 218, pp. 417–426

[3] LENSTRA A.K., VERHEUL E.R.: 'Selecting cryptographic key sizes', *J. Cryptol.*, 2001, **14**, pp. 255–293

[4] QUISQUATER J.J., SAMYDE D.: 'Electromagnetic analysis (EMA): measures and countermeasures for smart cards'. Smart Card Programming and Security, Int. Conf. on Research in Smart Cards (E-smart 2001), Cannes, France, 2001, vol. 2140, pp. 200–210

[5] GANDOLFI K., MOURTEL C., OLIVIER F.: 'Electromagnetic analysis: concrete results'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001), Paris, France, 2001, vol. 2162, pp. 251–261

[6] LEE J.Y., JUNG S.W., LIM J.: 'Detecting trapdoors in smart cards using timing and power analysis'. IFIP TC6/WG 6.1 Int. Conf.: Testing of Communicating Systems (TestCom 2005), Montreal, Canada, 2005, vol. 3502, pp. 275–288

[7] ANDERSON R.J., KUHN M.G.: 'Tamper resistance – a cautionary note'. The second USENIX workshop on electronic commerce, Oakland, CA, USA, 1996, vol. 2, pp. 1–11

[8] ANDERSON R.J., KUHN M.G.: 'Low cost attacks on tamper resistant devices'. Int. Workshop on Security Protocols, Paris, France, 1997, vol. 1361, pp. 125–136

[9] LIARDET P.Y., SMART NP: 'Preventing SPA/DPA in ECC systems using the Jacobi form'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001), Paris, France, 2001, vol. 2162, pp. 391–401

[10] JOYE M., QUISQUATER J.J.: 'Hessian elliptic curves and side-channel attacks'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001), Paris, France, 2001, vol. 2162, pp. 402–410

[11] OKEYA K., SAKURAI K.: 'Power analysis breaks elliptic curve cryptosystems even secure against the timing attack'. Int. Conf. in Cryptology in India (INDOCRYPT 2000), Calcutta, India, 2000, vol. 1977, pp. 178–190

[12] JOYE M., TYMEN C.: 'Protections against differential analysis for elliptic curve cryptography'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001), Paris, France, 2001, vol. 2162, pp. 377–390

[13] BRIER E., JOYE M.: 'Weierstrass elliptic curves and side-channel attacks'. Int. Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2002), Paris, France, 2002, vol. 2274, pp. 335–345

[14] MÖLLER B.: 'Parallelizable elliptic curve point multiplication method with resistance against side-channel attacks'. Int. Conf. on Information Security (ISC 2002), Sao Paulo, Brazil, 2002, vol. 2433, pp. 402–413

[15] CORON J.S.: 'Resistance against differential power analysis for elliptic curve cryptosystems'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 1999), Worcester, MA, USA, 1999, vol. 1717, pp. 292–302

[16] IZU T., TAKAGI T.: 'A fast parallel elliptic curve multiplication resistant against side channel attacks'. Int. Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2002), Paris, France, 2002, vol. 2274, pp. 280–296

[17] MÖLLER B.: 'Securing elliptic curve point multiplication against side-channel attacks'. Int. Conf. on Information Security (ISC 2001), Malaga, Spain, 2001, vol. 2200, pp. 324–334

[18] EDWARDS H.M.: 'A normal form for elliptic curves', *Bull. Am. Math. Soc.*, 2007, **44**, (3), pp. 393–422

[19] BERNSTEIN D.J., LANGE T.: 'Faster addition and doubling on elliptic curves'. Int. Conf. on Theory and Applications of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT 2007), Kuching, Malaysia, 2007, vol. 4833, pp. 29–50

[20] BERNSTEIN D.J., LANGE T., FARASHAHI R.R.: 'Binary Edwards curves'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2008), Washington, DC, USA, 2008, vol. 5154, pp. 244–265

[21] BADEL S.: 'MOS current-mode logic standard cells for high-speed low-noise applications'. PhD thesis, Swiss Federal Institute of Technology, Lausanne (EPFL), 2008

[22] JOYE M.: 'Highly regular right-to-left algorithms for scalar multiplication'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007), Vienna, Austria, 2007, vol. 4727, pp. 135–147

[23] HANKERSON D.R., VANSTONE S.A., MENEZES A.J.: 'Guide to elliptic curve cryptography' (Springer, 2004)

[24] LENSTRA A.K.: 'Cryptosystems with elliptic curves chosen by users'. US Patent 6,446,205, 3 September 2002

[25] CHUDNOVSKY D.V., CHUDNOVSKY G.V.: 'Sequences of numbers generated by addition in formal groups and new primality and factorization tests', *Adv. Appl. Math.*, 1986, **7**, (4), pp. 385–434

[26] LOPEZ J., DAHAB R.: 'Fast multiplication on elliptic curves over GF ($2^m$) without precomputation'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 1999), Worcester, MA, USA, 1999, vol. 1717, pp. 316–327

[27] COHEN H., MIYAJI A., ONO T.: 'Efficient elliptic curve exponentiation using mixed coordinates'. Int. Conf. on Theory and Applications of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT 1998), Beijing, China, 1998, vol. 1514, pp. 51–65

[28] REITWIESNER G.W.: 'Binary arithmetic', *Adv. Comput.*, 1960, **1**, pp. 231–308

[29] KOCHER P.: 'Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems'. Int. Cryptology Conf.: Advances in Cryptology (CRYPTO 1996), Santa Barbara, CA, USA, 1996, vol. 1109, pp. 104–113

[30] KOCHER P., JAFFE J., JUN B.: 'Differential power analysis'. Int. Cryptology Conference: Advances in Cryptology (CRYPTO 1999), Santa Barbara, CA, USA, 1999, vol. 1666, pp. 388–397

[31] CHARI S., RAO J.R., ROHATGI P.: 'Template attacks'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002), Redwood Shores, CA, USA, 2002, vol. 2523, pp. 13–28

[32] SCHINDLER W.: 'A combined timing and power attack'. Int. Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2002), Paris, France, 2002, vol. 2274, pp. 263–279

[33] CHEVALLIER-MAMES B., CIET M., JOYE M.: 'Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity', *IEEE Trans. Comput.*, 2004, **53**, (6), pp. 760–768

[34] BIEHL I., MEYER B., MULLER V.: 'Differential fault attacks on elliptic curve cryptosystems'. Int. Cryptology Conf.: Advances in Cryptology (CRYPTO 2000), Santa Barbara, CA, USA, 2000, vol. 1880, pp. 131–146

[35] YEN S.M., JOYE M.: 'Checking before output may not be enough against fault-based cryptanalysis', *IEEE Trans. Comput.*, 2000, **49**, (9), pp. 967–970

[36] JOYE M., YEN S.M.: 'The Montgomery powering ladder'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002), Redwood Shores, CA, USA, 2002, vol. 2523, pp. 291–302

[37] FOUQUE P.A., VALETTE F.: 'The doubling attack why upwards is better than downwards'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2003), Cologne, Germany, 2003, vol. 2779, pp. 269–280

[38] WALTER C.D.: 'Sliding windows succumbs to big mac attack'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001), Paris, France, 2001, vol. 2162, pp. 286–299

[39] CLAVIER C., CORON J.S., DABBOUS N.: 'Differential power analysis in the presence of hardware countermeasures'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2000), Worcester, MA, USA, 2000, vol. 1965, pp. 252–263

[40] AIGNER M., OSWALD E.: 'Power analysis tutorial'. Technical Report, Graz University of Technology (TU Graz), 2000

[41] PRAMSTALLER N., GURKAYNAK FK, HAENE S., KAESLIN H., FELBER N., FICHTNER W.: 'Towards an AES crypto-chip resistant to differential power analysis'. European Solid-State Circuits Conf. (ESSCIRC 2004), Leuven, Belgium, 2004, pp. 307–310

[42] GÜRKAYNAK F., OETIKER S., KAESLIN H., FELBER N., FICHTNER W.: 'Improving DPA security by using globally-asynchronous locally-synchronous systems'. European Solid-State Circuits Conf. (ESSCIRC 2005), Grenoble, France, 2005, pp. 407–410

[43] MESSERGES T.S.: 'Securing the AES finalists against power analysis attacks'. Int. Workshop on Fast Software Encryption (FSE 2000), New York, NY, USA, 2001, vol. 1978, pp. 150–164

[44] OSWALD E., MANGARD S., PRAMSTALLER N.: 'Secure and efficient masking of AES-A mission impossible'. Technical Report, Technical Report IAIK-TR 2003/11/1, 2004, http://eprint.iacr.org/




doi: 10.1049/iet-ifs.2009.0038

[45] TIRI K., AKMAL M., VERBAUWHEDE I.: 'A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards'. European Solid-State Circuits Conf. (ESSCIRC 2002), Florence, Italy, 2002, pp. 403–406

[46] TOPRAK Z., LEBLEBICI Y.: 'Low-power current mode logic for improved DPA-resistance in embedded systems'. Int. Symp. on Circuits and Systems (ISCAS 2005), Kobe, Japan, 2005, vol. 2, pp. 1059–1062

[47] REGAZZONI F., BADEL S., EISENBARTH T., *ET AL.*: 'A simulation-based methodology for evaluating the DPA-resistance of cryptographic functional units with application to CMOS and MCML technologies'. Int. Conf. on Embedded Computer Systems: Architectures, Modeling and Simulation (IC-SAMOS 2007), Samos, Greece, 2007, pp. 209–214

[48] AVIZIENIS A.: 'Signed-digit number representations for fast parallel arithmetic', *IRE Trans. Electron. Comput.*, 1961, **10**, (3), pp. 389–400

[49] TAKAGI N., YASUURA H., YAJIMA S.: 'High-speed VLSI multiplication algorithm with a redundant binary addition tree', *IEEE Trans. Comput.*, 1985, **34**, (9), pp. 789–796

[50] KORNERUP P.: 'Reviewing 4-to-2 adders for multi-operand addition', *J. VLSI Signal Process.*, 2005, **40**, (1), pp. 143–152

[51] KAIHARA M.E., TAKAGI N.: 'A hardware algorithm for modular multiplication/division', *IEEE Trans. Comput.*, 2005, **54**, (1), pp. 12–21

[52] TAKAGI N., YAJIMA S.: 'Modular multiplication hardware algorithms with a redundant representation and their application to RSA cryptosystem', *IEEE Trans. Comput.*, 1992, **41**, (7), pp. 887–891

[53] ORS S.B., BATINA L., PRENEEL B., VANDEWALLE J.: 'Hardware implementation of an elliptic curve processor over GF(p)'. 14th IEEE Int. Conf. on Application-Specific Systems, Architectures, and Processors (ASAP 2003), The Hague, The Netherlands, 2003, vol. 0, pp. 433–443

[54] SATOH A., TAKANO K.: 'A scalable dual-field elliptic curve cryptographic processor', *IEEE Trans. Comput.*, 2003, **52**, (4), pp. 449–460

[55] SOZZANI F., BERTONI G., TURCATO S., BREVEGLIERI L.: 'A parallelized design for an elliptic curve cryptosystem coprocessor'. Int. Symp. on Information Technology: Coding and Computing (ITCC 2005), Las Vegas, Nevada, USA, 2005, vol. 1, pp. 626–630

[56] ORS SB, BATINA L., PRENEEL B., VANDEWALLE J.: 'Hardware implementation of an elliptic curve processor over gf(p)'. IEEE Int. Conf. on Application-Specific Systems, Architectures, and Processors (ASAP 2003), The Hague, The Netherlands, 2003, pp. 433–443

[57] MENTENS N., SAKIYAMA K., BATINA L., PRENEEL B., VERBAUWHEDE I.: 'A side-channel attack resistant programmable pkc coprocessor for embedded applications'. Int. Conf. on Embedded Computer Systems: Architectures, Modeling and Simulation (IC-SAMOS 2007), Samos, Greece, 2007, pp. 194–200

[58] MCIVOR CJ, MCLOONE M., MCCANNY JV: 'Hardware elliptic curve cryptographic processor over gf(p)', *IEEE Trans. Circuits Syst. I Regul. Pap.*, 2006, **53**, (9), pp. 1946–1957

[59] SAKIYAMA K., MENTENS N., BATINA L., PRENEEL B., VERBAUWHEDE I.: 'Reconfigurable modular arithmetic logic unit supporting high-performance rsa and ecc over gf(p)', *Int. J. Electron.*, 2007, **94**, (5), pp. 501–514

[60] SAVAS E., KOC ÇK: 'Architectures for unified field inversion with applications in elliptic curve cryptography'. Ninth Int. Conf. on Electronics, Circuits and Systems (ICECS 2002), Dubrovnik, Croatia, 2002, vol. 3, pp. 1155–1158

[61] OZTURK E., SUNAR B., SAVAS E.: 'Low-power elliptic curve cryptography using scaled modular arithmetic'. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), Boston, MA, USA, 2004, vol. 3156, pp. 92–106

[62] SAVAS E., NASEER M., GUTUB A.A.A., KOC ÇK: 'Efficient unified Montgomery inversion with multibit shifting', *IEE Proc. – Comput. Dig. Tech.*, 2005, **152**, (4), pp. 489–498