Elliptic Curve Cryptography Engineering

ALESSANDRO CILARDO, LUIGI COPPOLINO, NICOLA MAZZOCCA, AND LUIGI ROMANO

Invited Paper

In recent years, elliptic curve cryptography (ECC) has gained widespread exposure and acceptance, and has already been included in many security standards. Engineering of ECC is a complex, interdisciplinary research field encompassing such fields as mathematics, computer science, and electrical engineering. In this paper, we survey ECC implementation issues as a prominent case study for the relatively new discipline of cryptographic engineering. In particular, we show that the requirements of efficiency and security considered at the implementation stage affect not only mere low-level, technological aspects but also, significantly, higher level choices, ranging from finite field arithmetic up to curve mathematics and protocols.

Keywords*—*Cryptographic engineering, cryptography, elliptic curves.

I. INTRODUCTION

After two decades of research and development, elliptic curve cryptography (ECC) [6] has now gained widespread exposure and acceptance, and has ultimately moved from being an interesting mathematical construction to a well-established public-key cryptosystem already included in numerous standards and adopted by an increasing number of companies. In fact, ECC is no longer new, and has withstood in the last years a great deal of cryptanalysis and a long series of attacks, which makes it appear as a mature and robust cryptosystem at present.

ECC has a number of advantages over other public-key cryptosystems, such as RSA, which make it an attractive alternative. In particular, for a given level of security, the size of the cryptographic keys and operands involved in the computation of EC cryptosystems are normally much shorter than other cryptosystems and, as the computational power available for cryptanalysis grows up, this difference gets more and more noticeable [6]. This is an advantageous condition especially for applications where resources such as memory and/or computing power are limited. Among these applications, the mobile computing area [26], with its ever increasing deployed base, is of extreme interest.

From the mathematical standpoint, an elliptic curve is the solution set to the bivariate polynomial equation $f(x, y) = 0$, where $f(x, y)$ is of total degree 3, and $f(x, y)$ is irreducible. In cryptography we consider particular equations and particular (finite) fields over which curves are defined. The points on the curve form a commutative group. What makes elliptic curves particularly attractive for cryptographic applications [8], [9] is that the discrete logarithm problem in elliptic curve groups is computationally hard. Moreover, it is "harder" than in groups previously considered, thereby allowing shorter key lengths.

However, the actual application of ECC, and the practical implementation of cryptosystem primitives in the real world constitute a complex and undoubtedly interdisciplinary research field, involving mathematics and computer science as well as electrical engineering. The different implementation aspects related to the EC cryptosystem are pictorially shown in Fig. 1. Alternative choices are available for building an EC-based cyptosystem at different levels, which roughly correspond to the horizontal layers shown in the figure. These range from the investigation of protocol robustness to software and hardware implementation of the underlying curve and finite fields arithmetics. Even though each of these aspects is often studied in isolation and constitutes a complex subject of interest in itself, it would be improper to say that layers of Fig. 1 are a hierarchical, unidirectional decomposition of the ECC engineering aspects. The two vertical requirements of implementation efficiency and implementation security make in fact the horizontal layers in the figure tightly interdependent. In general, the role of the implementation efficiency requirement (i.e., the possibility to achieve suitable levels of execution time, resources required, power consumed, costs,

Manuscript received October 29, 2004; revised April 16, 2005. This work was supported in part by the Italian National Research Council (CNR), in part by Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR), and in part by Regione Campania, within the framework of following projects: SP1 Sicurezza dei documenti elettronici, Gestione in sicurezza dei flussi documentali associati ad applicazioni di commercio elettronico, Centri Regionali di Competenza ICT, and Telemedicina.

The authors are with the Computer Science Department, University Federico II, Naples 80125, Italy (e-mail: acilardo@unina.it).

Digital Object Identifier 10.1109/JPROC.2005.862438

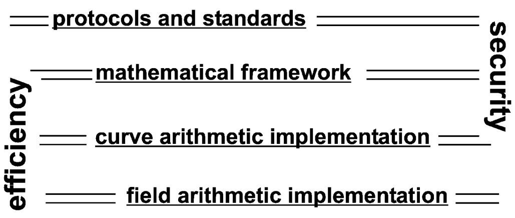

Fig. 1. Implementation aspects of ECC.

along with flexibility and reusability of design solutions) is rather clear: a mathematical construction with practical applications, such as a cryptographic algorithm, has no real interest, in an engineering sense, as long as methods for a feasible implementation are not available. That is probably the reason why public-key cryptography was not devised before the last decades of the 20th century. What is somewhat still implicit in the overall picture concerns the security aspect and is related to the fact that, surprisingly, implementation generates a significant, perhaps crucial portion of the real security risk. In fact, it is widely recognized that most of the threats inherent in real-world security infrastructures stem from *how* we perform cryptographic operations on secret data rather than from the intrinsic mathematical strength of the cryptosystem. As a consequence, architectural and implementation details, which normally take on a marginal role and seemingly just affect performance, are of crucial importance, as they could open the door to many real-world attacks in a number of nontrivial and often unforeseen ways.

In this paper, we show how these two aspects, which can be grouped under the label of *cryptographic engineering*, have a crucial role in steering design choices at all layers of Fig. 1 and reference several proposals found in the technical literature where the choices made at horizontal layers are driven "vertically" according to cryptographic engineering requirements. We start in Section II by reviewing the fundamental general requirements of cryptographic engineering in practical implementations. In Section III we give an overview of the existing implementation-specific security risks. Mathematical foundations and standards for ECC are introduced in Section IV. A review of implementation techniques and challenges for elliptic curve arithemetic and the underlying finite field arithmetic are given in Section V and Section VI, respectively. Section VII concludes the paper with some final remarks on elliptic curve cryptography design.

II. Cryptographic Engineering

Generally speaking, the strength of a public-key cryptosystem is directly related to the type of the one-way function it uses and the length of the cryptographic keys. With the computing power and the theoretic knowledge available today, we find that inverting a one-way function—the scalar multiplication for the case of EC cryptography—is a practically intractable problem. It is interesting to note that, for hard problems commonly used in cryptography, such as the RSA factoring problem and the elliptic curve discrete logarithm problem, this is just conjectured to be true, as no mathematical proof exists about the intrinsic complexity of such problems.

On the other hand, many specific algorithms exist to solve cryptographic hard problems used for public-key cryptosystems, and often their levels of computational complexity are pragmatically taken as a measure of the cryptosystem strength, even though it is not excluded that more efficient algorithms may exist. It has also been found that many feasible methods exist to attack particular instances of the hard problems (for example, the discrete logarithm problem over supersingular elliptic curves). The choice of the underlying mathematics, domain parameters, and key sizes is strongly influenced by such results when new cryptographic schemes and standards are defined. As long as all these choices are made appropriately, and no breakthroughs in number theory or computing technologies are achieved, the *key size* is the cryptosystem parameter that can be used for scaling the robustness of the cryptosystem over time or according to actual application needs. In fact, the key size usually affects the complexity of the underlying cryptographic hard problem, and thus determines the overall mathematical strength of the cryptosystem. In principle, we could obtain a cryptosystem as secure as we want by just increasing the key length at will.

However, in practical applications, there are some other considerations to do. First, in addition to being mathematically strong, the cryptosystem should be practically feasible. In fact, implementation efficiency depends on key sizes, as the cryptosystem security, although they follow different laws in general. A mathematically strong cryptosystem whose implementation requires prohibitive computation resources is of no practical utility.

In a wider sense, implementation efficiency could be seen as the property that allows a particular design solution, either software or hardware, to be used, and reused, in a scalable, modular, and flexible way. Most of these requirements are inherent in cryptographic algorithms and applications. In fact, the criteria of reusability and scalability are of fundamental importance for the design of cryptographic blocks, since the operand sizes, usually much larger than in normal applications, may significantly change depending on the required level of security or the specific cryptosystem. A design solution might turn out to be useless if not conceived for effectively respond to changes in operand sizes or algorithm parameters. This is mainly a problem for hardware architectures and is most likely to happen in the case of EC cryptography, since, from an implementation viewpoint, there are many tempting possibilities to exploit specific, parameter-dependent optimizations that work fine only for particular instances of the cryptosystem. Most recent proposals are deliberately oriented to emphasize scalability and reusability of cryptographic architectures, aiming at solutions that are parameter-independent, data length-independent, flexible and unified (in the sense that they allow heterogeneous operations, such as $GF(p)$ and $GF(2^m)$ arithmetics, to be computed by a single unit).

But there is another dimension in cryptographic engineering, distinct from mere implementation efficiency, that only recently has been fully recognized. This dimension has essentially to do with security and stems from the fact that, unfortunately, we cannot think of the *implementation* of a cryptographic primitive holding secret data (e.g., an integrated circuit card) as a perfect black box. Implementations do leak sensitive information, and such leakage might be far more significant than a possible unforeseen mathematical flaw in a cryptographic one-way function. In other words, cloning a smart card may be not as difficult as breaking the internal public-key algorithm, no matter how mathematically strong it is.

Both implementation efficiency and implementation security are essential aspects of cryptographic engineering, which we could indeed define as the science of translating cryptographic algorithms into feasible and secure design solutions.

One could easily recognize that cryptographic engineering is concerned with the vertical requirements shown in Fig. 1. Its role for real security applications is at least as important as mere theoretical concerns and it is now widely addressed in the technical literature, including the relatively new application of ECC. In the rest of this paper we will show how cryptographic engineering requirements influence design choices at all layers of Fig. 1, including definition of higher level algorithms and curve parameterizations.

III. Implementation-Specific Attacks

In this section we give an overview of the existing attacks that can be used for compromising tamper-resistant cryptographic devices based on implementation-specific characteristics rather than mathematical properties of the cryptosystem attacked. The study of implementation-specific attacks is a fundamental part of cryptographic engineering and is essential for the design of real-world cryptographic devices.

Implementation attacks are usually divided into *invasive* and *noninvasive* attacks. With invasive attacks the cryptographic device is physically tampered with using some special equipment. A general methodology for hardware devices consists in depackaging the chip and reverse engineering the silicon layout by means of optical microscopes and microprobing workstations. More advanced techniques are based on focused ion beam (FIB) workstations, electron-beam testers (EBT), and infrared lasers. Invasive attacks require a specialized laboratory and time-consuming reverse engineering operations and they always destroy the packaging of the card. On the other hand, they require very little initial knowledge and usually work with a similar set of techniques on a wide range of products [72].

Noninvasive, or side-channel attacks are based on the general idea of measuring some side-channel information on tamper-resistant devices, such as power consumption or timing, and attempting to infer secret keys from such information. Noninvasive attacks may be particularly dangerous. In fact, although the attack requires physical access to the device, usually it is not evident so that it is unlikely that the validity of the compromised data will be revoked before they are abused. In addition, side-channel attacks can be mounted with relatively inexpensive equipment. On the other hand, most noninvasive attacks require some preliminary knowledge of hardware and software aspects of the attacked device.

Noninvasive attacks can be divided into the following categories.

- *Timing Analysis Attacks*. Timing attacks [1] are based on measuring the time that the unit under attack requires to perform specific operations. They are based on the fact that, in straightforward implementations, many steps with different execution times may or may not be executed depending on the handled data, including secret keys.
- *Power Consumption Attacks*. Power attacks [2] require the interpretation of power consumption measurements collected during cryptographic operations and are based either on *Simple Power Analysis* (SPA) or *Differential Power Analysis* (DPA). SPA can be used to break cryptographic implementations in which the execution path depends on the data being processed, similar to timing attacks. The information leakage is usually tightly correlated to operand values and hamming weights. DPA attacks use statistical analysis and error correction techniques to extract information correlated to secret keys. Public-key algorithms, including EC algorithms, can be analyzed using DPA by correlating candidate values for intermediate results with power consumption measurements.
- *Electromagnetic Attacks*. Electromagnetic attacks [3], [4] are based on the same principle of power attacks applied to the analysis of electromagnetic radiation. They are classified into *Simple Electromagnetic Analysis* (SEMA) attacks and *Differential Electromagnetic Analysis* (DEMA) attacks.
- *Fault Analysis*. Fault Analysis techniques [11] consist in tampering with a device and making it perform some faulty computation that, when properly analyzed, may leak information about the secret parameters involved in the computation. Faults are induced by using heat, pressure, external electromagnetic fields, clock glitches, or power supply transients.

Many techniques exist to thwart implementation attacks. Most of them work at the technology level, including, for example, passivation layers, on-chip sensor meshes, radiation detectors, and power and clock frequency monitors. They are at a large extent decoupled from the characteristics of the particular cryptographic primitive being designed and, consequently, they are not specific to the case of ECC implementation. However, in recent years it has been recognized that significant vulnerabilities can stem from high-level algorithmic and architectural aspects as well, with special emphasis on side-channel attacks. Many countermeasures working at such levels have thus been proposed, and they are usually specific to the cryptographic algorithm being implemented. As the next sections will show, they multiply the available design choices and raise a number of interesting tradeoffs and issues for ECC engineering.

IV. ECC: Mathematics and Standards

A. ECC Mathematics

ECC is based on the discrete logarithm problem applied to the Abelian group formed by the points of an elliptic curve over a finite field. The essential security foundation of EC cryptosystems relies on the (supposed) absence of a subexponential algorithm for solving the discrete logarithm problem over cryptographic curves [6], [8], [9].

In terms of mathematical formulas, elliptic curves used in cryptography are defined over a finite field $\mathbb{K}$ by the Weierstraß equation

$$E_{/\mathbb{K}} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

The set of points $(x, y) \in \mathbb{K} \times \mathbb{K}$, along with the *point at infinity* $\mathcal{O}$, form an Abelian group, denoted with $E(\mathbb{K})$, where $\mathcal{O}$ is the identity element (i.e., $\forall P \in E(\mathbb{K}), P + \mathcal{O} = P$); the inverse of $P = (x_1, y_1)$ is $-P = (x_1, -y_1 - a_1x_1 - a_3)$; if $Q = -P$ then $P + Q = \mathcal{O}$; given $P = (x_1, y_1)$ and $Q = (x_2, y_2) \in E(\mathbb{K})$ with $Q \neq P$, $P + Q = (x_3, y_3)$ is computed as

$$\begin{aligned} x_3 &= \lambda^2 + a_1\lambda - a_2 - x_1 - x_2 \\ y_3 &= -(\lambda + a_1)x_3 - \mu - a_3 \end{aligned}$$

with

$$\lambda = \begin{cases} \left(\frac{y_1 - y_2}{x_1 - x_2}\right) & P \neq Q \\ \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & P = Q \end{cases}$$

and $\mu = y_1 - \lambda x_1$.

In cryptography we use two main families of elliptic curves, according to the base field $\mathbb{K}$ over which the curve is defined. In particular, for $\mathrm{Char}(\mathbb{K}) \neq 2, 3$, the general Weierstraß equation may be simplified to

$$E_{/\mathbb{K}} : y^2 = x^3 + ax + b \tag{1}$$

with $4a^3 + 27b^2 \not\equiv 0 \bmod p$. Taking $a_1 = a_2 = a_3 = 0$, $a_4 = a$, and $a_6 = b$ in the general Weierstraß equation, the sum of $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ (with $P \neq Q$) is given by $P + Q = (x_3, y_3)$ where

$$x_3 = \lambda^2 - x_1 - x_2 \quad \text{and} \quad y_3 = \lambda(x_1 - x_3) - y_1 \tag{2}$$

with $\lambda = (y_1 - y_2)/(x_1 - x_2)$ if $P \neq Q$ and with $\lambda = (3x_1^2 + a)/2y_1$ if $P = Q$.

For $\mathrm{Char}(\mathbb{K}) = 2$ the equation for (nonsupersingular) elliptic curves is given by

$$E_{/\mathbb{K}} : y^2 + xy = x^3 + ax^2 + b \tag{3}$$

with $b \neq 0$. Again, the sum of $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ (with $P \neq Q$) is given by $P + Q = (x_3, y_3)$ where

$$x_3 = \lambda^2 + \lambda + a + x_1 + x_2 \quad \text{and} \quad y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \tag{4}$$

with $\lambda = (y_1 - y_2)/(x_1 - x_2)$ if $P \neq Q$ and with $\lambda = x_1 + (y_1/x_1)$ if $P = Q$.

The number of points of $E_{/\mathbb{K}}$ is called the *curve order* and is denoted with $\#E_{/\mathbb{K}}$. The curve order can be computed in polynomial time by Schoof's algorithm [39] for general curves, and more efficiently by Satoh's technique [40] for curves of small characteristics, e.g., characteristic two.

Given a point $P = (x, y)$ on the curve and an integer k, a *scalar multiplication* on the curve is obtained as

$$kP = \underbrace{P + P + \ldots + P}_{k \text{ times}}. \tag{5}$$

The *order* of the point P is the smallest integer n such that $nP = \mathcal{O}$. Note that, if i and j are integers, then $iP = jP$ if and only if $i = j(\mathrm{mod}\, n)$.

Given a point P of order n and a point Q, the *elliptic curve discrete logarithm problem* (ECDLP) consists in finding an integer k, if any, such that $Q = kP$. The intractability of ECDL problem is the foundation of the EC cryptosystem. There are several algorithms to solve it, such as the Pollard-ρ method [63], and none of them is subexponential for appropriately chosen curves.

For general elliptic curves there are cases in which subexponential algorithms do exist, making special attacks possible. In particular, it is worth mentioning the Menezes/Okamato/Vanstone (MOV) reduction attack [31], the Smart/Araki-Satoh/Semaev attack [65], and the Weil Descent attack [54], all working under particular hypotheses on the attacked curves. These mathematical results are fundamental in that they have affected the design choices for protocols and standards used in real-world applications. For example, due to Menezes, Okamato, and Vanstone's result [31], supersingular curves are not used in applications. In fact, formulas for point operations could be significantly simplified for the case of such curves [60], particularly over $GF(2^m)$ and, hence, supersingular curves would lead to more efficient implementations. Nevertheless, as a result of the MOV reduction attack, it is possible to have subexponential attacks against supersingular elliptic curves, whereas the best-known attacks against nonsupersingular elliptic curves have still an exponential complexity. Standards consequently exclude the use of supersingular curves.

However, as mentioned in the introductory sections, efficiency and security measured only in terms of number theory metrics may not be enough. Cryptographic engineering requirements need to be taken into account. For example, the scalar multiplication (5), which involves the secret key k, is typically reduced to a sequence of additions and doublings, whose implementations typically behave in slightly different ways. As a consequence, an attacker could exploit sidechannel effects induced on a physical cryptodevice (possibly by passive, noninvasive observation techniques) to infer this sequence and extracts the secret value k while knowing nothing about the discrete logarithm problem. In order for such an attack to be practically mounted, it would be possible to exploit, for example, timing, power, or electromagnetic signals. Many ways exist to defeat such types of attacks, and they typically work at the implementation or technology level. But, additionally, these merely practical secutity threats can also imply such high-level responses as clever redefinitions of elliptic curve mathematics. These could be aimed, for example, at yielding elementary operations at a lower level, such as curve addition and doubling, that are indistinguishable when implemented. To this purpose, several authors suggested to use parameterizations for the elliptic curves that are different from the standard Weierstraß equation introduced at the beginning of this section. In [19], authors represent points with the Jacobi form as the intersection of two quadrics in $\mathbb{P}^3$. In [21] it is suggested to use the Hessian form. These parameterizations are not fully general, as opposed to the Weierstraß form. The Jacobi form has always a point of order 4 and the Hessian form a point of order 3, and thus the cardinality of the corresponding elliptic curve is a multiple of 4 and 3, respectively.

Other techniques used to improve implementation efficiency and implementation security, such as projective coordinates and unified addition formulas [23], work on the standard Weierstraß parameterization and will be considered in Section V.

B. *ECC Schemes and Standards*

As an introductory example of a cryptographic scheme based on the elliptic curve, we describe the EC Diffie–Hellman scheme used for key agreement. It is by far the simplest and the easiest to understand scheme, but it helps focusing on the role of the ECDL problem in a real application. Let Alice and Bob want to agree a shared key. As the set of domain parameters, they have agreed a specific curve and a base point P. Alice generates a random secret value x and sends the result of the scalar multiplication $x \cdot P$ to Bob over an insecure channel. Similarly, Bob generates a secret value y and sends $y \cdot P$ to Alice

$$\begin{array}{ccc} \text{Alice} & & \text{Bob} \\ x & \overset{x\cdot P}{\Rightarrow} & x \cdot P \\ y \cdot P & \overset{y\cdot P}{\Leftarrow} & y \end{array}.$$

Then, Alice and Bob compute

$$K_A = x \cdot (y \cdot P) = xy \cdot P$$
$$K_B = y \cdot (x \cdot P) = xy \cdot P$$

respectively, thereby sharing the secret value $K_A = K_B$. The problem of recovering $xy \cdot P$ given $x \cdot P$ and $y \cdot P$ is called the elliptic curve Diffie–Hellman problem (ECDHP). It is known that the ECDHP is "included" into the ECDLP, in the sense that if we can solve the ECDLP then we can solve the ECDHP.

Other fundamental EC-based cryptographic schemes include the Elliptic Curve Digital Signature Algorithm (EC-DSA), which extends the DSA to the case of elliptic curves, the Elliptic Curve Integrated Encryption Scheme (EC-IES), which basically works like static Diffie–Hellman followed by symmetric encryption, the Elliptic Curve Menezes, Qu, Vanstone (EC-MQV) scheme, which provides a mechanism for authenticated key exchange.

In the recent years, such elliptic curve-based cryptographic schemes have undergone a great deal of standardization efforts. In fact, EC cryptography is particularly prone to implementations that lack interoperability, due to the large number of parameters that must be chosen at different levels, from the underlying finite fields to the high-level cryptographic schemes. Among the other things, we need standardization in the definition of formats for representation of fields elements, curve points, cryptographic keys and so on. In addition, it is important to have a collection of recommended special curves and underlying finite fields in order to promote interoperability and favor particular technical choices that are appropriate from both a security and efficiency standpoint.

Among the main standards including ECC we cite the IEEE P1363 *Standard Specifications For Public-Key Cryptography* [41], the NIST *Digital Signature Standard* (DSS) [42], the ANSI *Public Key Cryptography for the Financial Services Industry* [43], [44], the OMA *Wireless Transport Layer Security Specification* (WTLS) [45], the SECG *SEC1* and *SEC2* standards [46], [47].

For example, the NIST standard includes a collection of recommended elliptic curves, private-key lengths, and underlying fields. It specifies how to represent field elements and provides for random-generated curves and selected curves. In particular, selected curves are Koblitz curves. The latter provision, along with several other recommendations, is made mainly for implementation efficiency reasons.

V. Elliptic Curve Arithmetic

This section isolates the problem of computing elliptic curve arithmetic, i.e., the second layer of Fig. 1. The basic operation for any EC cryptosystem is the scalar multiplication (5) from both a security and a performance point of view. The simplest way to implement it is to write kP as

$$\begin{aligned} kP &= \left(\sum_{i=0}^{l-1} k_i 2^i\right) P = (k_{l-1}2^{l-1}P) + \ldots + (k_1 2P) + (k_0 P) \\ &= 2\,(2\,(\ldots 2\,(2(k_{l-1}P) + k_{l-2}P) + \ldots) + k_1 P) + k_0 P \end{aligned}$$

and to use the following double-and-add algorithm:

Algorithm 1— *Double-and-Add Algorithm*

Given P, $k = (k_{l-1}, k_{l-2}, k_{l-3}, \ldots, k_0)_2$, computes $Q = kP$.

```
1. $R := \mathcal{O}$
2. for $j := l - 1$ downto $0$ do
3.     $R := 2R$
4.     if $(k_j = 1)$ then $R := R + P$
5. end for
6. return $R$
```

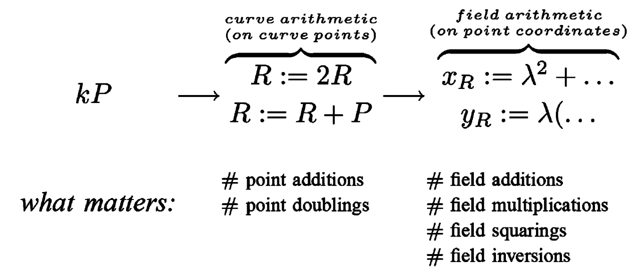

Fig. 2. Decomposition of elliptic curve arithmetic operations.

The double-and-add algorithm, as most alternative methods, reduce the scalar multiplication to additions ($R := R + P$) and doublings ($R := 2R$) of curve points. In turn, these are reduced to additions, multiplications, squarings, and inversions over the underlying finite fields according to (2) for $GF(p)$ and (4) for $GF(2^m)$ as shown in Section IV. Pictorially, the relation between the different arithmetics is as shown in Fig. 2.

It is important to note that x_R and y_R, denote *affine* coordinates and are just one of different ways for representing a curve point R. By using other representations it is possible to obtain different formulas for additions and doublings and thus a different count of additions, multiplications, squarings, and inversions in field arithmetic (involved in the right-hand part of the chain in Fig. 2). This is very important for implementation efficiency reasons, as the different field operations have in general very different levels of complexity. In particular, the weight of field additions is generally negligeable, and inversion is usually much more complex than multiplication. For example, with affine coordinates the formula for adding a curve point over $GF(2^m)$ (4) is reduced to one field inversion, two multiplications and one squaring (field addition has marginal influence). Instead, we could use *Jacobian* projective coordinates [6], [41]. These are obtained from affine coordinates (x, y) as $(X, Y, Z) = (\lambda^2 x, \lambda^3 y, \lambda)$, where λ is a nonzero element of the underlying field. From (X, Y, Z) the affine coordinates are computed as $(x, y) = (X/Z^2, Y/Z^3)$. Jacobian projective coordinates yield formulas for point addition and doubling that require more multiplications and squarings but do not require inversion at all, except for initial and final conversions. For example, for the same addition of a curve point over $GF(2^m)$ considered before, Jacobian projective coordinates require 11 field multiplications and four field squarings. Other types of coordinates have been proposed and studied, including standard [6], Lopez–Dahab [37], and mixed coordinates [38].

As projective coordinates trade a number of field multiplications for one inversion, whether to resort to projective coordinates or to the affine ones critically depends on inversion/multiplication weight ratio in the actual implementation. There are several ways to optimize field inversions and multiplications for different fields and representations (see Section VI), and these lead to various inversion/multiplication weight ratios, which makes impossible to establish a unique optimal design choice. In usual cases it turns out to be more convenient to resort to projective or mixed coordinates.

The double-and-add Algorithm 1 requires about $\log_2 k$ point doublings (step 3) and $(\log_2 k)/2$ point additions (step 4). The latter contribution is essentially related to the number of nonzero digits in k. A different way to speed up the computation of the scalar multiplication $k \cdot P$ is to work on the left-hand side of the chain in Fig. 2 by *recoding* the integer k in such a way that the number of nonzero digits in it be lower. This makes the sequence of operations necessary to build the kth multiple of P shorter than the conventional double-and-add method, usually at the price of doing some precomputation and storing some additional quantities. For example, it is possible to group consecutive bits of k and work on bit "windows" rather than scanning single bits as in the case of the double-and-add method of Algorithm 1. This is done with the so-called m-ary method, which partitions the binary expansion of the integer k into fixed-length, r-bit windows ($m = 2^r$). It requires the precomputation of $2^{r-1} - 1$ general points additions, and takes $\log_2 k$ doublings and about $(\log_2 k)/r$ additions. A more efficient technique exploits *sliding windows* in the representation of k, which are similar to the m-ary method but work on nonfixed partitions of k [71]. In particular, with a *signed* sliding window technique we can represent k as a sum of signed components: $k = \sum_{i=0}^{n-1} k_i 2^{e_i}$ with $e_{i+1} - e_i \leq r$ and $k_i \in \{\pm 1, \pm 3, \ldots, \pm 2^{r-1} - 1\}$. The use of the signed representation and thus the possibility of subtracting multiples of P to partial results allows us to further reduce the number of point additions/subtractions and is particular appealing for elliptic curves, since in this case the negation comes practically for free.

Similar principles apply to the so-called nonadjacent form (NAF) [16], [48], [50], which represents k as $\sum_{j=0}^{n-1} k_j 2^j$ with $k_j \in \{0, +1, -1\}$. It has been shown that each integer admits a uniquely determined NAF, which is the signed binary representation of minimal Hamming weight and has an expected density of 1/3 [51]. The binary NAF technique can be generalized to the case of *width-w* nonadiacent form [52].

It should be noted that all these techniques tend to reduce the number of additions/subtractions (step 4 in Algorithm 1) rather than the number—or the weight—of doublings (step 3). Particular advantages for implementation of curve arithmetic come from special curves over $GF(2^m)$, known as Koblitz curves [49], that are obtained from (3) with $b = 1$ and $a \in \{0, 1\}$. The main property of such curves is that if (x, y) is a point on a Koblitz curve, then (x^2, y^2) is also a point on the curve and in addition we have $\tau(x, y) = (x^2, y^2)$ (Frobenius map) where $\tau^2 - (-1)^{1-\alpha}\tau + 2 = 0$. It follows that the multiplication of a point on the curve by the complex number τ can simply be realized with the squaring of the x and y coordinates of the point, which in normal basis representation is as simple as a cyclic shift of the bits of the operands. We can thus represent the integer k with radix τ [48] that implies replacing the doubling in the double-and-add method with a less expensive multiplication by τ.

A similar idea, suitable for certain classes of curves over $GF(p)$, is the GLV method [64]. The method works for elliptic curves over $GF(p)$ possessing an endomorphism ϕ whose characteristic polynomial has small coefficients. The scalar k is first decomposed into two "small" integers k_0 and k_1 (called a balanced length-two representation of k). The point-multiplication $k \cdot P$ is then computed as $k \cdot P = k_0 \cdot P + k_1 \cdot \phi(P)$, taking advantage of the simultaneous multiple exponentiation technique known as the "Shamir's trick."

Other techniques exist to optimize the computation of scalar multiplication for specific classes of curves. For example, the technique used in [37] works for $GF(2^m)$ and speeds up the computation of $2^i \cdot P$ used in window methods. Authors in [61] propose some formulas for computing repeated doublings in projective coordinates that work in combination with window methods. In [28] authors use some techniques first introduced in [29] (including the so-called Montgomery ladder) to speed up the scalar multiplication process. In particular, they exploit the fact that, given two elliptic points P_1 and P_2 whose difference is fixed and equal to P, the x coordinate of $P_1 + P_2$ can be computed in terms of the x coordinates of P, P_1, and P_2. In [62] the *halving* operation (i.e., given P, the computation of Q such that $2Q = P$) is proposed to replace the doubling operation in the scalar multiplication process.

Projective coordinates, sliding windows, nonadiacent form, etc. are all techniques working on the curve arithmetic layer of Fig. 1 for speeding up implementation. However, again, we have to point out that design choices made at this level do not affect only implementation performance but also implementation security. In fact, the double-and-add Algorithm 1 is a representative example of how naïve approaches to computing sensitive operations may lead to implementations that potentially leak secret data [5]. The formulas for doubling and adding points on a Weierstraß elliptic curve entail different groups of operations, and a simple power analysis (i.e., a simple side-channel analysis using power consumption as the side channel) will produce different power traces that may reveal the bits of k. One way to circumvent the leakage can be to have an algorithm that behaves regurarly whatever the processed data. This is done, for example, in [24], where a double-and-add-*always* approach is taken: here, a (possibly meaningless) addition is always performed at step 4 of Algorithm 1 irrespective of the value of k_j. In [25] the Montgomery ladder is used for scalar multiplication mainly as a measure against power and fault attacks for its intrinsically regular behavior. In [23] the addition formulas are rewritten in such a way that point addition and point doubling are no longer distinguishable. In [27] the authors propose a method which is applicable on general curves over prime-order finite-fields and works without precomputed points. Among the most promising techniques for defeating side-channel effects we cite randomization techniques, which are particularly effective for differential attacks [5]. Broadly speaking, such techniques work by obfuscating algorithm inputs and operands in such a way that the implementation behavior is weakly related to the actual values of secret data [7]. Note that these techniques do not necessarily yield implementations optimized for performance.

VI. Finite Fields Arithmetic

Elliptic curve operations are ultimately reduced to operations over finite fields (see Fig. 2). As a consequence, the implementation efficiency of ECC crucially depends on the performance of finite field computations.

A finite field, also called a Galois field (GF), is just a field with a finite number of elements (field order). A fundamental result on the theory of finite fields states that there exists a finite field of order q if and only if q is a prime power. For each prime power q there is essentially only one field of order q.

There is a large variety of choices for finite fields to use as the underlying fields in ECC. Popular choices include fields of the type $GF(p)$ where p is a large prime. Arithmetic for such fields can be implemented as the usual modular arithmetic and can take advantage of a large number of techniques already found in the literature and previously developed for other cryptosystems such as DSA and RSA. In particular, for general primes the most efficient implementation technique is based on Montgomery arithmetic [55], which uses a special representation of numbers to obtain efficient implementation of modular multiplication. However, since the number p is a constant in the case of an elliptic curve cryptosystem, it is common in standards to choose primes of a special form that allows a drastic optimization for modular reduction. These include Mersenne numbers and generalized Mersenne (GM) numbers [32]. All major standards include GM numbers [42]–[47].

Much important are also nonprime fields, i.e., fields of the form $GF(p^m)$ where p is a prime and $m > 1$. In particular, standards use binary fields $GF(2^m)$. Special nonprime fields have also been proposed, including optimized extension fields (OEFs) [53] With OEFs the field parameters can be chosen such that they are a good match for the processor on which the field arithmetic is to be implemented. In particular, it is often an advantage to choose an OEF $GF(p^m)$ such that the prime p can be represented within one register of the target processor. From a security standpoint, it should be pointed out that there exist fields that are "weaker" than others. This is the case for composite fields of characteristic two [54].

Representation of field elements is particularly important for implementation efficiency. While elements of $GF(p)$ are naturally reviewed as integer numbers, there are many ways of representing elements in $GF(p^m)$ fields. We discuss here the usual case of binary finite fields (i.e., fields of the form $GF(2^m)$). Such fields can be viewed as a vector space of dimension m over $GF(2)$. Thus, there is a set of elements $\{\alpha_0, \alpha_1, \ldots, \alpha_{m-1},\}$ in $GF(2^m)$ such that each $a \in GF(2^m)$ can be written uniquely in the form $a = \sum_{i=0}^{m-1} a_i \alpha_i$, where $a_i \in 0, 1$. The set $\{\alpha_0, \alpha_1, \ldots, \alpha_{m-1},\}$ is called a *basis* of $GF(2^m)$ and defines a representation of the elements in $GF(2^m)$. In particular, the so-called *polynomial basis* are defined by irreducible polynomials $f(x) = x^m + \sum_{i=0}^{m-1} f_i x^i$ (where $f_i \in \{0, 1\}$) of degree m over $GF(2)$. With a polynomial basis, field elements can be thought of as binary polynomials of degree less than m and the field operations can be carried out formally by using the conventional polynomial addition and multiplication modulo $f(x)$. From an implementation viewpoint, it is a good choice to use irreducible polynomials of low weight (usually trinomials or pentanomials) with low order nonleading terms, as most standards do (see for example [42]). A different possibility is to use *normal bases*, i.e., bases of the form $\{\beta, \beta^2, \ldots, \beta^{2m-1}\}$, where $\beta \in GF(2^m)$. It is always possible to find such a basis [33]. The representations based on normal bases have the computational advantage that squaring a field element is as simple as shifting its components. Addition is carried out by bitwise XOR-ing elements' components, as in the polynomial basis case. On the other hand, multiplication is in general a more complicated operation, unless a special class of normal bases, called Gaussian normal bases, is used [41]. A discussion of normal bases and a characterization of their existence are given in [34], [35]. Other types of bases have been proposed for addressing specific performance issues, for example the problem of efficiently computing field inversion/division [36].

There is a plethora of research works dealing with the problem of efficiently computing arithmetic in finite fields. In general, there are proposals for implementation of arbitrary operations as well as techniques that are highly optimized for specific parameters such as the irreducible polynomial used for representing $GF(2^m)$ elements in the polynomial basis. This is especially true of field multiplication [56], [57], [68]–[70]. Many different proposals exist for field inversion/division [36], [59], [66], [67]. In particular, there is an increasing interest in devising *unified* hardware arithmetic architectures, allowing heterogeneous field operations, such as multiplication or inversion in both $GF(p)$ and $GF(2^m)$, to be performed by single hardware blocks. Much emphasis is also put on scalability, i.e., the ability of reusing or replicating a unit in order to obtain an arbitrary precision independently of the data path precision of the single unit [57]. Indeed, in addition to computation speed, flexibility and scalability play a fundamental role for the vertical requirement of implementation efficiency. For example, based on the extension of Montgomery algorithm to the $GF(2^m)$ case [58], authors in [57] achieve scalable and unified hardware architectures implementing multiplication in $GF(p)$ and $GF(2^m)$. Similarly, in [59] a scalable and unified architecture is proposed for computing Montgomery inversion in $GF(p)$ and $GF(2^m)$. Further works about unification include [73], [75], and [74]. In [74], in particular, a change of representation in $GF(2^m)$ is proposed to enable unified arithmetic based on Montgomery operations. Implementation of finite fields arithmetic has also important implications on power consumption, which is a crucial aspect in resource-constrained environments such as smart-cards and handheld devices. Much attention has been recently paid to power efficient approaches for implementation of finite field arithmetic [76].

VII. Concluding Remarks on ECC Design

The design of elliptic curve based cryptographic systems implies a number of nontrivial challenges and tradeoffs.

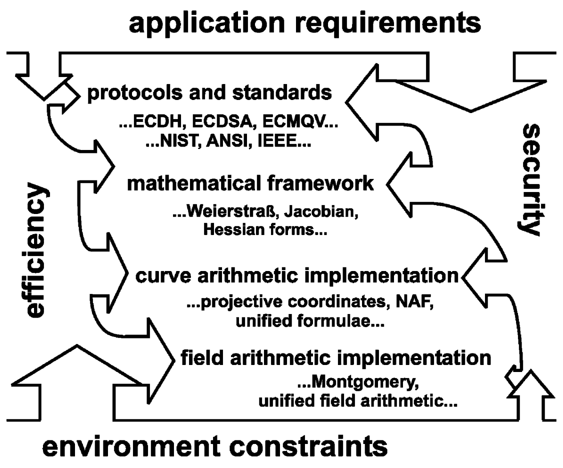
application requirements
protocols and standards
...ECDH, ECDSA, ECMQV...
...NIST, ANSI, IEEE...
mathematical framework
...Weierstraß, Jacobian, Hessian forms...
curve arithmetic implementation
...projective coordinates, NAF, unified formulae...
field arithmetic implementation
...Montgomery, unified field arithmetic...
efficiency
security
environment constraints

Fig. 3. ECC design map.

Fig. 3 summarizes the factors involved in the design process and their impact on the layers of the ECC engineering map. First of all, the target computing environment is likely to influence significantly the design process. This is especially true of ECC, since it finds a natural application in resource-constrained systems, such as PDAs, mobile phones, and smart cards. The overall available resources—processing power, memory, power consumption—often determine the functionality of the system designed and, in that respect, they conflict with application requirements. For example, on a smart card it is reasonable to implement ECC based signature and encryption, but it is not so for curve generation.

On the other hand, the characterization of the target application usually influences the way such resources are exploited. In fact, different applications require different cryptographic schemes and in turn involve the underlying elliptic curve operations in different ways. As a result, spending large efforts for improving the execution time of a specific operation may or may not make sense. In addition, the application scope and expected lifecycle have a considerable impact on implementation efforts toward flexibility and scalability. It is crucial to assess the level of flexibility necessary in order for the designed system to respond to changes in parameters/algorithms/schemes that are likely to occur during its lifecycle. Similar observations hold for security requirements. Most security measures sketched in the previous sections at all layers of the ECC design map have their costs in terms of either execution time, area/memory consumption, or technology processes and may or may not be worth depending on the commercial value of the application developed and the financial impact of the security risk. This analysis should also take into account the costs of known and potential attacks, generally not negligeable, as well as the fact that such attacks must be separately mounted for each cryptodevice to break.

Once the impact of computing environment constraints and application requirements is assessed, a large number of design choices must be addressed at each layer of Fig. 3, and, correspondingly, many different tradeoffs can be recognized. These include the following.

- *Implementation efficiency versus implementation security*. As shown in the previous sections, at each layer of the ECC design map there are a number of implementation options that are specifically related to the problem of obtaining secure designs. For example, for the implementation of elliptic curve arithmetic there are many available algorithms in the technical literature (see Section V) that are deliberately conceived for thwarting side-channel attacks. There measures often entail nonoptimal time/area performance and, in some cases, limit the generality of the implemented solutions. While a slight penalty in performance is usually acceptable in favor of security, compromising the flexibility of the design solution would in general require a deeper cost/benefit analysis.
- *Time-to-market versus speed/flexibility/security*. The time-to-market is perhaps the major factor that impacts the commercial value of a design solution. Special measures enabling implementation efficiency, flexibility, and security are all likely to increase development times with respect to straightforward approaches. On the other hand, generic off-the-shelf solutions (IP cores, software programs) could not meet such specific requirements, or otherwise they could entail additional costs.
- *Execution time versus area/memory requirements*. The usual time versus area tradeoffs hold at many different layers of ECC design map, including among the other things double-and-add versus windowing methods, scalar versus parallel field multiplicator, etc.
- *Speed versus flexibility*. As mentioned in the introductory sections, implementation of ECC offers many tempting possibilities of exploiting parameter-specific optimizations or particular standard prescriptions. For example, the implementation of field multiplication takes advantage of particular irreducible polynomials used for representation, often suggested by standards such as NIST [42].

The latter point highlights the impact of standard compliancy on the design process. In fact, standards usually contain many implementation-aware provisions, and many standard parameters, such as curve type/coefficients, field order, etc., are defined with hardware or software implementation implications in mind. Indeed, the implementation of ECC standards is probably very different from the implementation of general ECC algorithms.

A design choice that takes on a particular importance for elliptic curve based systems is the hardware/software partitioning. Where the bound between hardware and software should be placed significantly depends, again, from the specific class of parameters adopted. For instance, implementation of $GF(2^m)$ arithmetic drastically benefits from hardware acceleration due to its carryless nature, while for $GF(p)$ arithmetic there are many efficent techniques for software implementation. Whether to use dedicated hardware blocks or not has considerable implications on design budgets, time-to-market, flexibility, scalability, and even security alongside mere time performace. Many research works and commercial proposals suggest that generic hardware support for field arithmetic and software control for the higher layers could turn out to be a profitable choice for usual applications, as it enables speed optimization of the basic mathematical operations (e.g., field multiplications/inversions) while leaving room for flexibility, scalability and "algorithmic" security of the overall design.

Below the layers of Fig. 3 are the design choices made at the technology level. These include low-level protection mechanisms (randomised clocking, noise generation, sensor meshes, dual-rail logic etc.) and radical technology choices such as the use of application-specific ICs (ASICs) or field-programmable gate arrays (FPGAs), with significant implications on security, flexibility, speed, costs. We have not discussed the technology level here, since it is mostly orthogonal to the classes of security applications designed above it. Conventional techniques for production of cryptographic devices and previous research investigations about protection methods and implementation technologies (e.g., ASICs versus FPGAs) still apply to the case of ECC.

These concluding remarks highlight the fact that, although ECC is now a mature and widely accepted alternative to traditional public-key cryptosystems, engineering of ECC is still an open research field and constitutes a complex, interdisciplinary subject. This paper surveyed the most significant aspects of ECC engineering at different levels and showed that implementation efficiency and implementation security—the fundamental goals of cryptographic engineering—are vertical requirements affecting not only mere technological aspects considered at the implementation stage, but also, significantly, choices at the higher levels, ranging from finite field arithmetic up to curve equations and algorithms, and security standards.

Acknowledgment

The authors would like to thank J. Großschädl for his comments and valuable suggestions.

References

[1] P. C. Kocher, "Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems," in *Advances in Cryptology—CRYPTO'96*, N. Koblitz, Ed.. Heidelberg, Germany: Springer-Verlag, 1996, vol. 1109, Lecture Notes in Computer Science, pp. 104–113.

[2] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO'99*, M. Wiener, Ed. Heidelberg, Germany: Springer-Verlag, 1999, vol. 1666, Lecture Notes in Computer Science, pp. 388–397.

[3] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded Systems—CHES 2001*, Ç. K. Koç, D. Naccache, and C. Paar, Eds. Heidelberg, Germany: Springer-Verlag, 2001, vol. 2162, Lecture Notes in Computer Science, pp. 251–261.

[4] J.-J. Quisquater and D. Samyde, "ElectroMagnetic Analysis (EMA): Measures and counter-measures for smard cards," in *Smart Card Programming and Security (E-smart 2001)*. Heidelberg, Germany: Springer-Verlag, 2001, vol. 2140, Lecture Notes in Computer Science, pp. 200–210.

[5] M. Joye, "Elliptic curves and side-channel analysis," *ST J. Syst. Res.*, vol. 4, no. 1, pp. 17–21, Feb. 2003.

[6] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*. Cambridge, U.K.: Cambridge University Press, 1999, vol. 265, London Mathematical Society Lecture Note Series.

[7] ——, *Advances in Elliptic Curve Cryptography*. Cambridge, U.K.: Cambridge University Press, 2005, vol. 317, London Mathematical Society Lecture Note Series.

[8] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.,* vol. 48, no. 177, pp. 203–209, 1987.

[9] V. S. Miller, "Use of elliptic curves in cryptography," in *Advances in Cryptology—CRYPTO'85*, H. C. Williams, Ed. Heidelberg, Germany: Springer-Verlag, 1986, vol. 218, Lecture Notes in Computer Science, pp. 417–426.

[10] R. Anderson*, Security Engineering*. New York: Wiley, 2001.

[11] I. Biehl, B. Meyer, and V. Muller, "Differential fault attacks on elliptic curve cryptosystems," in *Advances in Cryptology—CRYPTO 2000*, M. Bellare, Ed. Heidelberg, Germany: Springer-Verlag, 2000, vol. 1880, Lecture Notes in Computer Science, pp. 131–146.

[12] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in Cryptology—CRYPTO '97*, B. Kaliski, Ed. Heidelberg, Germany: Springer-Verlag, 1997, vol. 1294, Lecture Notes in Computer Science, pp. 513–525.

[13] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Advances in Cryptology—EUROCRYPT '97*, W. Fumy, Ed. Heidelberg, Germany: Springer-Verlag, 1997, vol. 1233, Lecture Notes in Computer Science, pp. 37–51.

[14] M. Joye, A. K. Lenstra, and J.-J. Quisquater, "Chinese remaindering based cryptosystems in the presence of faults," *J. Cryptol.,* vol. 12, no. 4, pp. 241–245, 1999.

[15] J. F. Dhem, F. Koeune, P. A. Leroux, P. Mestré, J.-J. Quisquater, and J. L. Willems, "A practical implementation of the timing attack," in *Smart Card Research and Applications (CARDIS 98)*, J.-J. Quisquater and B. Schneier, Eds. Heidelberg, Germany: Springer-Verlag, 2000, vol. 1820, Lecture Notes in Computer Science, pp. 167–182.

[16] G. W. Reitwiesmer, "Binary arithmetic," *Adv. Comput.,* vol. 1, pp. 231–308, 1960.

[17] O. Eğecioğlu and Ç. K. Koç, "Exponentiation using canonical recoding," in *Proc. 1990 Bilkent Int. Conf. New Trend in Communication, Control, and Signal Processing* pp. 188–194.

[18] K. Okeya, H. Kurumatani, and K. Sakurai, "Elliptic curves with the montgomery form and their cryptographic applications," in *Public Key Cryptography (PKC'00)*, J.-J. Quisquater and B. Schneier, Eds. Heidelberg, Germany: Springer-Verlag, 2000, vol. 1751, Lecture Notes in Computer Science, pp. 238–257.

[19] P. Liardet and N. Smart, "Preventing SPA/DPA in ECC system using the Jacobi form," in *Cryptographic Hardware and Embedded Systems—CHES 2001*. Heidelberg, Germany: Springer-Verlag, 2001, vol. 2162, Lecture Notes in Computer Science, pp. 401–411.

[20] O. Billet and M. Joye, "The Jacobi model of an elliptic curve and side-channel analysis" Cryptology ePrint Archive, Rep. 2002/125, 2002 [Online]. Available: http://eprint.iacr.org/2002/125.pdf

[21] M. Joye and J. Quisqiater, "Hessian elliptic curves and side-channel attacks," in *Cryptographic Hardware and Embedded Systems—CHES 2001*. Heidelberg, Germany: Springer-Verlag, 2001, vol. 2162, Lecture Notes in Computer Science, pp. 412–420.

[22] N. Smart, "The Hessian form of an elliptic curve," in *Cryptographic Hardware and Embedded Systems—CHES 2001*. Heidelberg, Germany: Springer-Verlag, 2001, vol. 2162, Lecture Notes in Computer Science, pp. 118–125.

[23] E. Brier and M. Joye, "Weierstrass elliptic curves and side-channel attacks," in *Public Key Cryptography (PKC'02)*. Heidelberg, Germany: Springer-Verlag, 2002, vol. 2274, Lecture Notes in Computer Science, pp. 335–345.

[24] J. S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," in *Cryptographic Hardware and Embedded Systems—CHES 1999*, Ç. K. Koç and C. Paar, Eds. Heidelberg, Germany: Springer-Verlag, 1999, vol. 1717, Lecture Notes in Computer Science, pp. 292–302.

[25] M. Joye and S. M. Yen, "The Montgomery powering ladder," in *Cryptographic Hardware and Embedded Systems—CHES 2002*, B. S. Kaliski, Jr., Ç. K. Koç, and C. Paar, Eds. Heidelberg, Germany: Springer-Verlag, 2003, vol. 2523, Lecture Notes in Computer Science, pp. 291–302.

[26] K. Lauter, "The advantages of elliptic curve cryptography for wireless security," *IEEE Wireless Commun.,* vol. 11, no. 1, pp. 62–67, Feb. 2004.

[27] T. Izu and T. Takagi, "A fast parallel elliptic curve multiplications resistant against side channel attacks," in *Public Key Cryptography*, D. Naccache and P. Paillier, Eds. Heidelberg, Germany: Springer-Verlag, 2002, vol. 2274, Lecture Notes in Computer Science, pp. 280–296.

[28] J. López and R. Dahab, "Fast multiplication on elliptic curves over $GF(2^m)$ without precomputation," in *Cryptographic Hardware and Embedded Systems*, Ç. K. Koç and C. Paar, Eds. Heidelberg, Germany: Springer-Verlag, 1999, vol. 1717, Lecture Notes in Computer Science, pp. 316–327.

[29] P. L. Montgomery, "Speeding the Pollard and elliptic curve methods of factorization," *Math. Comput.,* vol. 48, no. 177, pp. 243–264, Jan. 1987.

[30] K. Okeya and K. Sakurai, "Power analysis breaks elliptic curve cryptosystems even secure against the timing attack," in *Progress in Cryptology—INDOCRYPT 2000*, B. Roy and E. Okamoto, Eds. Heidelberg, Germany: Springer-Verlag, 2000, vol. 1977, Lecture Notes in Computer Science, pp. 178–190.

[31] A. Menezes, T. Okamoto, and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field," *IEEE Trans. Inf. Theory,* vol. 39, no. 5, pp. 1639–1646, Sep. 1993.

[32] J. Solinas, "Generalized Mersenne numbers" Dept. Combinatorics and Optimizations, Univ. Waterloo, Tech. Rep. CORR99-06, 1999 [Online]. Available: http://www.cacr.math.uwaterloo.ca/

[33] R. J. McEliece*, Finite Fields for Computer Scientists and Engineers*. Norwell, MA: Kluwer, 1987.

[34] D. Ash, I. Blake, and S. Vanstone, "Low complexity normal bases," *Discrete Appl. Math.,* vol. 25, pp. 191–210, 1989.

[35] R. Mullin, I. Onyszchuk, S. Vanstone, and R. Wilson, "Optimal normal bases in $GF(p^n)$," *Discrete Appl. Math.,* vol. 22, pp. 149–161, 1989.

[36] M. Hasan, "Double-basis multiplicative inversion over $GF(2^m)$," *IEEE Trans. Comput.,* vol. 47, no. 9, pp. 960–970, Sep. 1998.

[37] J. Lopez and R. Dahab, "Improved algorithms for elliptic curve arithmetic in $GF(2^m)$," in *Selected Areas in Cryptography*. Heidelberg, Germany: Springer-Verlag, 1998, vol. 1556, Lecture Notes in Computer Science, pp. 201–212.

[38] H. Cohen, A. Miyaji, and T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates," in *Asiacrypt'98*. Heidelberg, Germany: Springer-Verlag, 1998, vol. 1514, Lecture Notes in Computer Science, pp. 51–65.

[39] R. Schoof, "Elliptic curves over finite fields and the computation of square roots mod p," *Math. Comp.,* vol. 44, pp. 483–494, 1985.

[40] T. Satoh, "The canonical lift of an ordinary elliptic curve over a finite field and its point counting," *J. Ramanujan Math. Soc.,* vol. 15, no. 4, pp. 247–270, 2000.

[41] *Standard specifications for public-key cryptography*, IEEE P1363 [Online]. Available: http://grouper.ieee.org/groups/1363/P1363/index.html

[42] *Digital signature standard (DSS)*, FIPS PUB 186-2, National Institute of Standards and Technology (NIST), 2000.

[43] *Public key cryptography for the financial services industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, ANSI X9.62-1998, American National Standards Institute.

[44] *Public key cryptography for the financial services industry: Key agreement and key transport using elliptic curve cryptography*, ANSI X9.63-2001, American National Standards Institute.

[45] Open Mobile Alliance, "Wireless transport layer security specification (WTLS)," 2001.

[46] Standards for Efficient Cryptography Group (SECG), "SEC 1: Elliptic curve cryptography," 2000.

[47] Standards for Efficient Cryptography Group (SECG), "SEC 2: Recommended elliptic curve domain parameters," 2000.

[48] J. A. Solinas, "Improved algorithms for arithmetic on a family of elliptic curves," in *Proc. Advances in Cryptography, Crypto-97* 1997, pp. 357–371.

[49] N. Koblitz, "CM-curves with good cryptographic properties," in *Proc. Advances in Cryptography, Crypto-91* 1992, pp. 279–287.

[50] D. Gordon, "A survey of fast exponentiation methods," *J. Algorithms,* vol. 27, pp. 129–146, 1998.

[51] F. Morain and J. Olivos, "Speeding up the computations on an elliptic curve using addition-subtraction chains," *RAIRO Inf. Theory,* vol. 24, pp. 531–543, 1990.

[52] J. Solinas, Department of Combinatorics and Optimizations, University of Waterloo, “Improved algorithms for arithmetic on a family of elliptic curves (revised)” Tech. Rep. CORR99-06, 1999 [Online]. Available: http://www.cacr.math.uwaterloo.ca/

[53] D. Bailey and C. Paar, “Optimal extension fields for fast arithmetic in public-key algorithms,” in *CRYPTO’98*. Heidelberg, Germany: Springer-Verlag, 1998, vol. 1462, Lecture Notes in Computer Science, pp. 472–485.

[54] P. Gaudry, F. Hess, and N. P. Smart, “Constructive and destructive facets of weil descent on elliptic curves,” *J. Cryptol.,* vol. 15, no. 1, pp. 19–46, 2002.

[55] P. L. Montgomery, “Modular multiplication without trial division,” *Math. Comput.,* vol. 44, no. 170, pp. 519–521, 1985.

[56] A. Halbutoğullari and Ç. K. Koç, “Mastrovito multiplier for general irreducible polynomials,” *IEEE Trans. Comput,,* vol. 49, no. 5, pp. 503–518, 2000.

[57] A. F. Tenca, E. Savas, and Ç. K. Koç, “A design framework for scalable and unified multipliers in $GF(p)$ and $GF(2^m)$,” *International Journal of Computer Research,* vol. 13, no. 1, pp. 68–83, 2004.

[58] Ç. K. Koç and T. Acar, “Montgomery multiplication in $GF(2^k)$,” *Designs, Codes and Cryptography,* vol. 14, no. 1, pp. 57–69, 1998.

[59] A. A.-A. Gutub, A. F. Tenca, E. Savas, and Ç. K. Koç, “Scalable and unified hardware to compute Montgomery inverse in $GF(p)$ and $GF(2^n)$,” Cryptographic Hardware and Embedded Systems (CHES) 2002B. S. Kaliski, Jr., Ç. K. Koç, and C. Paar, Eds., 4th International Workshop, Springer Verlag, LNCS, vol. 2523, pp. 484–499, 2002.

[60] A. Menezes and S. Vanstone, “Elliptic curve cryptosystems and their implementation,” *J. Cryptol.,* pp. 209–224, 1993.

[61] K. Itoh, M. Takenaka, N. Torii, S. Temma, and Y. Kurihara, “Fast implementation of public-key cryptography on a DSP TMS320C6201,” in *Cryptographic Hardware and Embedded Systems (CHES99).* Heidelberg, Germany: Springer-Verlag, 1999, vol. 1717, Lecture Notes in Computer Science, pp. 61–72.

[62] E. W. Knudsen, “Elliptic scalar multiplication using point halving,” in *Asiacrypt’99*. Heidelberg, Germany: Springer-Verlag, 1999, vol. 1716, Lecture Notes in Computer Science, pp. 135–149.

[63] R. Gallant, R. Lambert, and S. Vanstone, “Improving the parallelized Pollard lambda search on anomalous binary curves,” *Math. Comput.,* vol. 69, pp. 1699–1705, 2000.

[64] R. P. Gallant, R. J. Lambert, and S. A. Vanstone, “Faster point multiplication on elliptic curves with efficient endomorphisms,” in *Crypto-2001*. Heidelberg, Germany: Springer-Verlag, 2001, vol. 2139, Lecture Notes in Computer Science, pp. 190–200.

[65] N. Smart, “The discrete logarithm problem on elliptic curves of trace one,” *J. Cryptol.,* vol. 12, pp. 193–196, 1999.

[66] M. A. Hasan, “Efficient computation of multiplicative inverses for cryptographic applications,” in *Proc. 5th IEEE Symp. Computer Arithmetic* 2001, pp. 66–72.

[67] T. Itoh and S. Tsujii, “A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases,” *Inf. Comput.,* vol. 78, pp. 171–177, 1988.

[68] E. Mastrovito, “VLSI architectures for computation in Galois Fields,” Ph.D. dissertation, Dept. Electr. Eng., Linköping University, Linköping, Sweden, 1991.

[69] C. Paar and P. S. Rodriguez, “Fast arithmetic architectures for public-key algorithms over Galois fields $GF((2^n)^m)$,” in *Advances in Cryptography, EUROCRYPT ’97*. Heidelberg, Germany: Springer-Verlag, 1997, vol. 1233, Lecture Notes in Computer Science, pp. 363–378.

[70] L. Song and K. Parhi, “Efficient finite fields serial/parallel multiplication,” in *Proc. Int. Conf. Application Specific System Architectures and Processors* 1996, pp. 72–82.

[71] K. Koyama and Y. Tsuruoka, “Speeding up elliptic cryptosystems by using a signed binary window method,” in *Advances in Cryptology, CRYPTO ’92*. Heidelberg, Germany: Springer-Verlag, 1993, vol. 740, Lecture Notes in Computer Science, pp. 345–357.

[72] O. Kömmerling and M. G. KuhnDesign, “Principles for tamper-resistant smartcard processors,” presented at the USENIX Workshop Smartcard Technology (Smartcard ’99), Chicago, IL.

[73] A. Satoh and K. Takano, “A scalable dual-field elliptic curve cryptographic processor,” *IEEE Trans. Comput.,* vol. 52, no. 4, pp. 449–460, Apr. 2003.

[74] A. Cilardo, A. Mazzeo, and N. Mazzocca, “A representation of elements in F_{2^m} enabling unified field arithmetic for elliptic curve cryptography,” *IEEE Electron. Lett.,* vol. 41, no. 14, Jul. 2005.

[75] J. Großschädl, “A bit-serial unified multiplier architecture for finite fields $GF(p)$ and $GF(2^m)$,” in *Cryptographic Hardware and Embedded Systems*. Heidelberg, Germany: Springer-Verlag, 2001, vol. 2162, Lecture Notes in Computer Science, pp. 202–219.

[76] J. Großschädl and G. A. Kamendje, “Low-power design of a functional unit for arithmetic in finite fields $GF(p)$ and $GF(2^m)$,” in *Information Security Applications*. Heidelberg, Germany: Springer Verlag, 2003, vol. 2908, Lecture Notes in Computer Science, pp. 227–243.

Alessandro Cilardo received the five-year degree in computer science engineering, *magna cum laude*, from the University of Naples Federico II, Naples, Italy, in 2003. He is currently working toward the Ph.D. degree at the same university.

His current research interests include algorithms and architectures for cryptographic engineering, with particular emphasis on public-key cryptosystems. His research activities also include investigation of architectures for digital time stamping services.

Mr. Cilardo received the 2003 Federcomin-AICA Award for the best thesis in the field of Information and Communication Technology. He is the creator of the IP-SIM project, one of the winners of the eGate Contest 2005, sponsored by Axalto, SUN Microsystems, and ST Microelectronics.

Luigi Coppolino received the degree in computer engineering from the University of Naples Federico II, Naples, Italy, in 2003 and the University Master Degree in “Real Time, Reliable and Secure Computing for Industrial Applications” from the Seconda Università degli Studi of Naples Federico II in 2004. He is currently working toward the Ph.D. degree at the University of Naples Federico II.

His research interests include fault tolerant and secure systems and embedded systems.

Nicola Mazzocca received the M.Sc. degree in electronic engineering and the Ph.D. degree in computer engineering from the University of Naples Federico II, Naples, Italy.

He is currently a Professor of High-Performance and Reliable Computing at the Computer and System Engineering Department of the University of Naples Federico II, Italy. He has authored over 200 papers in international journals, books, and international conferences in the field of computer architecture, reliable and secure systems, distributed systems, and performance evaluation in high-performance systems. His research activities include methodologies and tools for design/analysis of distributed systems; techniques for modeling and analysis of distributed heterogeneous systems and communication networks; secure and real-time systems; distributed control applications; models and tools for configuration and performance evaluation of distributed, heterogeneous systems and communication networks; dedicated parallel architectures. In the context of such activities, he took part in numerous research projects, funded by the Italian Ministry of University and Research (MIUR), the National Research Council (CNR), the Agenzia Spaziale Italiana (ASI), and the European Union. He cooperated with a number of Italian and foreign institutions, including the Universities of Turin, Florence, and Parma, the CNR, the University of Illinois at Urbana-Champaign, Caltech, the University of Sheffield, and the Jet Propulsion Lab (JPL).

Luigi Romano received the M.Sc. Degree in electronic engineering and the Ph.D. degree in computer engineering from the University of Naples Federico II, Naples, Italy, in 1994 and 1999, respectively. He is currently an Associate Professor at the Department for Technologies of the University of Naples Parthenope. In 1996–1997, he was appointed as a Visiting Scholar at the Centre for Reliable and High-Performance Computing (CRHC), of the University of Illinois, Urbana-Champaign. In 1997–1998, he was again appointed to CRHC, as a Visiting Researcher. He spent in total about 18 months at CRHC, doing research with Prof. R. K. Iyer. His research interests are dependability and security of computer systems and networks, embedded systems, and middleware and Web technologies.

Dr. Romano is a member of the International Program Committees for several international conferences in the field of dependable and secure computing, including the International Symposium on Dependable Systems and Networks (DSN), the Symposium on Reliable Distributed Systems (SRDS), and the International Symposium on High Assurance Systems Engineering (HASE).