
Universidad
Carlos III de Madrid

e-Archivo
Institutional Repository

This is a postprint version of the following published document:

Gayoso Martínez V., González-Manzano L., Martín Muñoz A. (2018) Secure Elliptic Curves in Cryptography. In: Daimi K. (eds) Computer and Network Security Essentials. Springer, Cham
Available in https://doi.org/10.1007/978-3-319-58424-9_16

© Springer International Publishing AG 2018

Secure Elliptic Curves in Cryptography

Victor Gayoso Martínez, Lorena González-Manzano, and Agustín Martín Muñoz

16.1 Introduction

In 1985, Neal Koblitz [24] and Victor Miller [28] independently suggested using elliptic curves defined over finite fields for implementing different cryptosystems. This branch of public-key cryptography is typically known as Elliptic Curve Cryptography (ECC), and its security is based on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP), which is considered to be more difficult to solve than the Integer Factorization Problem (IFP) used by RSA or the Discrete Logarithm Problem (DLP) which is the basis of the ElGamal encryption scheme [11, 16, 24]. Since the inception of ECC, elliptic curves have been typically represented in what is called the short Weierstrass form (which is described in Sect. 16.2).

One of the most important aspects when working with secure elliptic curves is how they are generated. Even though some standards include several sample curves or even the description of the procedures for generating them (e.g., X9.63 [2], IEEE 1363 [21], or NIST FIPS 186 [30]), in most cases the information contained in those standards has important limitations, such as the lack of clarity in the selection procedure regarding the seeds and prime numbers involved or the insufficient explanation for some of the requirements specified in the procedure.

V. Gayoso Martínez (✉) • A. Martín Muñoz
Institute of Physical and Information Technologies (ITEFI), Spanish National Research Council (CSIC), Madrid, Spain
e-mail: victor.gayoso@iec.csic.es; agustin@iec.csic.es

L. González-Manzano
Computer Security Lab (COSEC), Universidad Carlos III de Madrid, Leganés, Madrid, Spain
e-mail: lgmanzan@inf.uc3m.es

In this scenario, in the early 2000s a working group called ECC Brainpool focused on this topic and completed a first set of recommendations in 2005 [10] for elliptic curves in the short Weierstrass form. Five years later, the Brainpool specification was revised and published as a Request for Comments (RFC) [25]. The Brainpool initiative was considered to be the first international effort with the goal of producing a truly transparent curve generation scheme, and the curves suggested in its specification were initially considered to be secure without any hint of doubt.

Several years later, researchers Daniel Bernstein and Tanja Lange published an analysis in which they reviewed the existing elliptic curve generation mechanisms, including the one devised by Brainpool. In their website SafeCurves [6], they compared not only the strength of the curve parameters and the soundness of what they called "ECC security" (basically the strength against rho attacks [34] and transfers of the ECDLP to other fields where the DLP is easier to solve [18–20, 27], the class number associated with the trace of the curve [12], and the rigidity of the definition of the curve parameters) but also what they termed "ECDLP security," a concept in which they included the resistance to attacks based on the Montgomery ladder [29], the strength of the associated twisted curves [8], the completeness of the addition formulas [5], and the indistinguishability of elliptic curve points from random binary strings [9]. The main result of that analysis was that all the schemes included in the standards overlooked some aspects of the ECDLP security and, for that reason, required to increase the complexity of the implementations in such a way that it opened the door to some types of side channel attacks [6].

As a solution, Bernstein and Lange decided to propose new curves different to those provided by previous specifications. Going one step further, they evaluated 20 curves obtained from different sources, showing that only a small subset of curves fulfilled all their security requirements. That subset is composed by elliptic curves in the Edwards and Montgomery formats (both of them introduced in Sect. 16.2).

However, from the point of view of availability, Montgomery and Edwards curves have not been popular choices so far, and in that respect traditional curves in the short Weierstrass form are the dominant option in both hardware and software implementations. In addition to that, the extra security offered by Edwards and Montgomery curves could affect the performance of the point operations which are the core of the scalar-multiplication operation (the product of a point of the elliptic curve by an integer, an operation needed in any protocol involving elliptic curves).

Regarding the resistance of elliptic curve cryptosystems to quantum computing, at the time of writing this contribution the National Security Agency (NSA) has confirmed the status of transition algorithms to some ECC cryptosystems while new cryptographic systems that are secure against both quantum and classical computers are defined [31, 33]. As it is not clear when those new systems will be available, it is safe to state that ECC will continue to be used at least in the near future.

This chapter presents to the reader the different types of elliptic curves used in Cryptography as well as the Brainpool procedure. The contribution is completed with the examination of the proposal regarding secure elliptic curves represented by the SafeCurves initiative.

16.2 Elliptic Curves

In this section, the reader is presented with the mathematical description of elliptic curves, as well as the specific details of elliptic curves described in the short Weierstrass form and the Edwards and Montgomery formats.

16.2.1 Definition

An elliptic curve defined over a field $\mathbb{F}$ is a cubic, non-singular curve whose points $(x, y) \in \mathbb{F} \times \mathbb{F}$ verify the following equation, known as the Weierstrass equation:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$ and $\triangle \neq 0$, where $\triangle$ is the discriminant of E that can be computed as follows [26]:

$$\begin{aligned}
\triangle &= -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6, \\
d_2 &= a_1^2 + 4a_2, \\
d_4 &= 2a_4 + a_1a_3, \\
d_6 &= a_3^2 + 4a_6, \\
d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2.
\end{aligned}$$

An elliptic curve point is singular if and only if the partial derivatives of the curve equation are null at that point. The curve is said to be singular if it possesses at least a singular point, while it is non-singular if it does not have any such points.

The nonhomogeneous Weierstrass equation can also be expressed in the following homogeneous form [12]:

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3.$$

This equation defines a curve which includes a special point called the point at infinity, which is typically represented as $\mathcal{O} = [0 : 1 : 0]$ and that has no correspondence with any point of the nonhomogeneous form. However, this point is very important as it works as the identity element of the addition operation when working with Weierstrass and Montgomery elliptic curves.

16.2.2 Elliptic Curves Over Finite Fields

Most cryptosystems defined over elliptic curves use only the following finite fields $\mathbb{F}_q$ with $q = p^m$ elements: prime fields $\mathbb{F}_p$ (where p is an odd prime number and

$m = 1$) and binary fields $\mathbb{F}_{2^m}$ (where m can be any positive integer). However, due to a combination of license issues and security concerns [4], prime fields have been favored in the latest specifications at the expense of binary fields (see, for example, Brainpool [25], NSA Suite B [32], or BSI TR-03111 [11]). Following that criterion, this chapter focuses on elliptic curves defined over prime fields. In this type of curves, the term key length must be interpreted as the number of bits needed to represent the prime number p.

When using prime fields, the order (i.e., the number of points) of any elliptic curves is finite. In this context, the order of a point P of the elliptic curve is the minimum nonzero value n such that $n \cdot P = \mathcal{O}$, where $n \cdot P$ is the scalar multiplication of the point P by the number n (i.e., $P + P + \cdots + P$, where P appears n times).

A point G is said to be a generator if it is used to generate either all the points of the curve or a subset of those points. For security reasons, only generators whose order is a prime number are used in Cryptography.

Given a curve and a generator, the term cofactor refers to the result of dividing the number of points of the curve by the order of the generator. Most standards only allow curves whose cofactor is either 1 or a small number like 2, 3, or 4.

16.2.2.1 Weierstrass Curves

The peculiarities of prime fields allow to simplify the general Weierstrass equation, obtaining in the process what is called the short Weierstrass form represented as $y^2 = x^3 + ax + b$, where $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$.

As in the case of the general Weierstrass equation, the identity element of the short Weierstrass form is the point at infinity $\mathcal{O}$, while the opposite element of a point $P = (x, y)$ is the point $-P = (x, -y)$. Adding two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ such that $P_1 \neq \pm P_2$ produces a point $P_3 = (x_3, y_3)$ whose coordinates can be computed as follows [7]:

$$x_3 = \frac{(y_2 - y_1)^2}{(x_2 - x_1)^2} - x_1 - x_2, \qquad y_3 = \frac{(2x_1 + x_2)(y_2 - y_1)}{x_2 - x_1} - \frac{(y_2 - y_1)^3}{(x_2 - x_1)^3} - y_1.$$

In comparison, when $P_1 = P_2$ it is necessary to use an alternative addition formula, so in this case the point $P_3 = 2P_1$ obtained through the doubling operation has the following coordinates [7]:

$$x_3 = \frac{(3x_1^2 + a)^2}{(2y_1)^2} - 2x_1, \qquad y_3 = \frac{(3x_1)(3x_1^2 + a)}{2y_1} - \frac{(3x_1^2 + a)^3}{(2y_1)^3} - y_1.$$

16.2.2.2 Edwards Curves

Edwards curves were introduced by Harold M. Edwards in [15], though during the last decade slightly different equations have been given that name. The first

expression related to Edwards curves was its *normal form* $x^2 + y^2 = c^2(1 + x^2y^2)$, where the neutral element of the addition operation is the point $(0, c)$ [15]. Edwards showed that any elliptic curve could be transformed into its normal form if the finite field used by the curve is algebraically closed. If that was not the case, then only a small fraction of elliptic curves could be transformed into the normal form using the original finite field [5, 8].

A few months later, Bernstein and Lange presented a variant using the *generalized form* $x^2 + y^2 = c^2(1 + dx^2y^2)$, where $cd(1 - dc^4) \not\equiv 0 \pmod{p}$ [5]. The goal of Bernstein and Lange was to increase the number of curves that could be converted into the Edwards form using the original finite field. In addition to the previous definition, an Edwards curve is said to be *complete* if d is not a square in $\mathbb{F}_p$ (i.e., if d is not a quadratic residue in $\mathbb{F}_p$), which allows to use only one addition operation for any pair of points, avoiding the case of exceptional points. If $c \neq 0$, $d \neq 0$, d is not square, and $dc^4 \neq 1$, then the coordinates of P_3 can be computed as follows for any pair of points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$:

$$x_3 = \frac{x_1y_2 + y_1x_2}{c(1 + dx_1x_2y_1y_2)}, \qquad y_3 = \frac{y_1y_2 - x_1x_2}{c(1 - dx_1x_2y_1y_2)}.$$

Bernstein and Lange also proved in [5] that all curves in the generalized form are isomorphic to curves defined by means of the equation $x^2 + y^2 = 1 + dx^2y^2$, which is the equation typically associated with regular *Edwards curves* in the literature. If $d \in \mathbb{F} - \{0, 1\}$, then in this type of curves the point $(0, 1)$ is the identity element of the addition operation, the point $(0, -1)$ has order 2, and the points $(1, 0)$ and $(-1, 0)$ have order 4.

One year later, Bernstein et al. proposed another generalization, producing as a result *twisted Edwards curves*, defined by the equation $ax^2 + y^2 = 1 + dx^2y^2$ [8]. The addition formula for twisted Edwards curves is the same as in the case of Edwards curves in the generalized form, where $c = 1$ in the case of twisted Edwards curves.

Theoretically, it would be possible to work with a more general model given by the equation $ax^2 + y^2 = c^2(1 + dx^2y^2)$. However, as the curves defined according to that model are always isomorphic to twisted Edwards curves, in practice it is not used [8].

The most interesting characteristic of complete twisted Edwards curves is that the equations for both adding two points P_1 and P_2 such that $P_1 \neq \pm P_2$ and doubling a point are exactly the same. Moreover, it is not necessary to implement any logic for detecting if the points to be added are such that $P_2 = -P_1$, as the Edwards addition equations also take into account that circumstance. This feature is important when defining the countermeasures to some type of side channel attacks.

16.2.2.3 Montgomery Curves

Montgomery curves conform to the equation $By^2 = x^3 + Ax^2 + x$, where $B(A^2 - 4) \not\equiv 0 \pmod{p}$. As in the case of Weierstrass curves, the identity element in Montgomery

curves is the point at infinity $\mathcal{O}$, while the opposite element of $P = (x, y)$ is the point $-P = (x, -y)$. The addition of two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ such that $P_1 \neq \pm P_2$ is the point $P_3 = (x_3, y_3)$ with the following coordinates [7]:

$$x_3 = \frac{B(y_2 - y_1)^2}{(x_2 - x_1)^2} - A - x_1 - x_2, \quad y_3 = \frac{(2x_1 + x_2 + A)(y_2 - y_1)}{x_2 - x_1} - \frac{B(y_2 - y_1)^3}{(x_2 - x_1)^3} - y_1.$$

Unlike Edwards curves, it is necessary to use different equations for the doubling operation, so in this case the coordinates of the point $P_3 = 2P_1$ can be computed as follows [7]:

$$x_3 = \frac{B(3x_1^2 + 2ax_1 + 1)^2}{(2By_1)^2} - A - 2x_1,$$
$$y_3 = \frac{(3x_1 + A)(3x_1^2 + 2Ax_1 + 1)}{2By_1} - \frac{B(3x_1^2 + 2Ax_1 + 1)^3}{(2By_1)^3} - y_1.$$

16.2.3 Transforming Formulas

Complete twisted Edwards curves $ax^2 + y^2 = 1 + dx^2y^2$ are birationally equivalent to Montgomery curves $By^2 = x^3 + Ax^2 + x$, where two curves are birationally equivalent if their fields of rational functions are isomorphic [15] (when referring to projective non-singular curves, the term *birationally equivalent* simply means that the curves are isomorphic). This means that every Montgomery curve can be expressed as a twisted Edwards curve, and vice versa [8]. In order to transform a complete twisted Edwards elliptic curve into a Montgomery elliptic curve as displayed in the previous section, it is necessary to use the following equivalence formulas [6]:

$$A = 2\frac{a + d}{a - d}, \qquad B = \frac{4}{a - d}.$$

In this way, a curve point (x_E, y_E) which belongs to an Edwards curve can be converted to a point (x_M, y_M) of the associated Montgomery curve, where the equations for obtaining (x_M, y_M) are as follows:

$$x_M = \frac{1 + y_E}{1 - y_E}, \qquad y_M = \frac{x_M}{x_E}.$$

Besides, in order to transform a Montgomery elliptic curve into the short Weierstrass form, it is necessary to use the following equivalences [6]:

$$a = \frac{3 - A^2}{3B^2}, \qquad b = \frac{2A^3 - 9A}{27B^3}$$

In this specific case, a curve point (x_M, y_M) belonging to a Montgomery curve can be converted to a point (x_W, y_W) of the associated short Weierstrass curve, where the transforming equations are the following ones:

$$x_W = \frac{x_M + \frac{A}{3}}{B}, \qquad y_W = \frac{y_M}{B}.$$

It is important to notice that, given the number and nature of the field operations needed in each case, elliptic curves in the short Weierstrass form have a better performance than the elliptic curves expressed in the Edwards and Montgomery forms when computing scalar multiplications, which is the basic operation when dealing with elliptic curves [14].

16.3 Brainpool

Even though elliptic curve cryptographic protocols are well defined in standards from ANSI [2, 3], IEEE [21, 22], ISO/IEC [23], NIST [30], and other organizations, it is usually the case that the elliptic curve parameters that are necessary to operate those protocols are offered to the reader without a complete and verifiable pseudo-random generation process. Some of the most important limitations detected across the main cryptographic standards regarding the processes for generating elliptic curves suitable for Cryptography are the following [10]:

- The seeds used to generate the curve parameters are typically chosen ad hoc.
- The primes that define the underlying prime fields have a special form aimed at facilitating efficient implementations.
- The parameters specified do not cover in all the cases key lengths adapted to the security levels required nowadays.

In this scenario, a European consortium of companies and government agencies led by the Bundesamt für Sicherheit in der Informationstechnik (BSI) was formed in order to study the aforementioned limitations and produce their recommendations for a well-defined elliptic curve generation procedure. The group was named ECC Brainpool (henceforth simply Brainpool), and, apart from the BSI, some of the most relevant companies and public institutions that took part in the effort were G&D, Infineon Technologies, Philips Electronics, Gemplus (now part of Gemalto), Siemens, the Technical University of Darmstadt, T-Systems, Sagem Orga, and the Graz University of Technology.

In 2005, Brainpool delivered the first version of a document entitled "ECC Brainpool standard curves and curve generation" [10], which was revised and published as an RFC memorandum in 2010, the "Elliptic Curve Cryptography (ECC) Brainpool standard curves and curve generation" [25]. The following sections present the main characteristics of the Brainpool procedure.

16.3.1 Key Length

As mentioned before, the Brainpool procedure only manages elliptic curves defined over prime fields expressed in the short Weierstrass form. The key lengths allowed by Brainpool are 160, 192, 224, 256, 320, 384, and 512 bits [25, p. 6].

16.3.2 Seed Generation

The seeds used in Brainpool are generated in a systematic and comprehensive way. These seeds have been obtained as the first 7 substrings of 160 bits each of the number $\pi \cdot 2^{1120} = Seedp160|| \ldots ||Seedp512||Remainder$, where $||$ denotes the concatenation operator [25, p. 24].

16.3.3 Seed to Candidate Conversion

Brainpool uses SHA-1 [25, p. 22] during the process of finding candidates for the parameters p, a, and b, as it can be observed in Fig. 16.1, where L represents the bit length of p. Even though it is not recommended to use SHA-1 as a hashing function in security environments (e.g., digital signatures), it is important to note that in this context it is only used for generating candidate values.

As the output of the hashing function SHA-1 is 160 bits, and for different curves the length of the resulting parameters must be necessarily different, Brainpool performs a loop concatenating several SHA-1 outputs until the concatenated number has the proper bit length [25, pp. 22 and 24].

In addition to that, Brainpool uses two functions to generate the candidates, one for p and another for a and b. Those functions are very similar; in fact, the only difference is that the most significant bit of a and b is forced to be 0 [25, p. 24]. Given that another requirement states that the most significant bit of p must be 1 [25, p. 23], this implies that the values a and b generated are such that $a, b < p$.

16.3.4 Validation of Parameters a and b

In Brainpool, once the algorithm has determined the value of p, it starts searching the proper values for the elliptic curve parameters a and b, as it can be seen in Fig. 16.2. When a candidate pair is found, the resulting curve is tested against the security requirements. In case the curve is rejected, both a and b are discarded, starting a new search for a proper pair [25, p. 25].

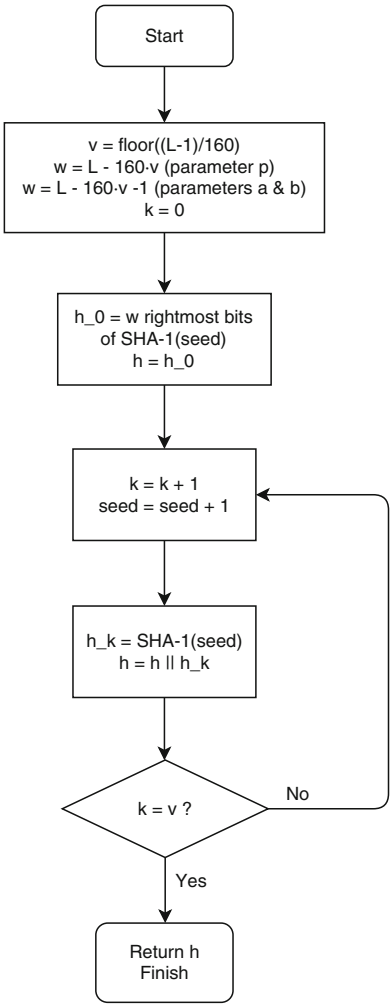

Fig. 16.1 Generation of candidates for parameters p, a, and b in Brainpool

16.3.5 Cofactors

In order to generate cryptographically strong elliptic curves, it is necessary to compute the number of points of the elliptic curve and to determine if that value is a prime number or if it has a small cofactor. In this regard, the Brainpool specifications only accept curves whose number of points is a prime number [25, p. 6]. This means that the Brainpool curves cannot be transformed into the twisted Edwards or Montgomery forms, as in those types of curves the number of points is always divisible by 4.

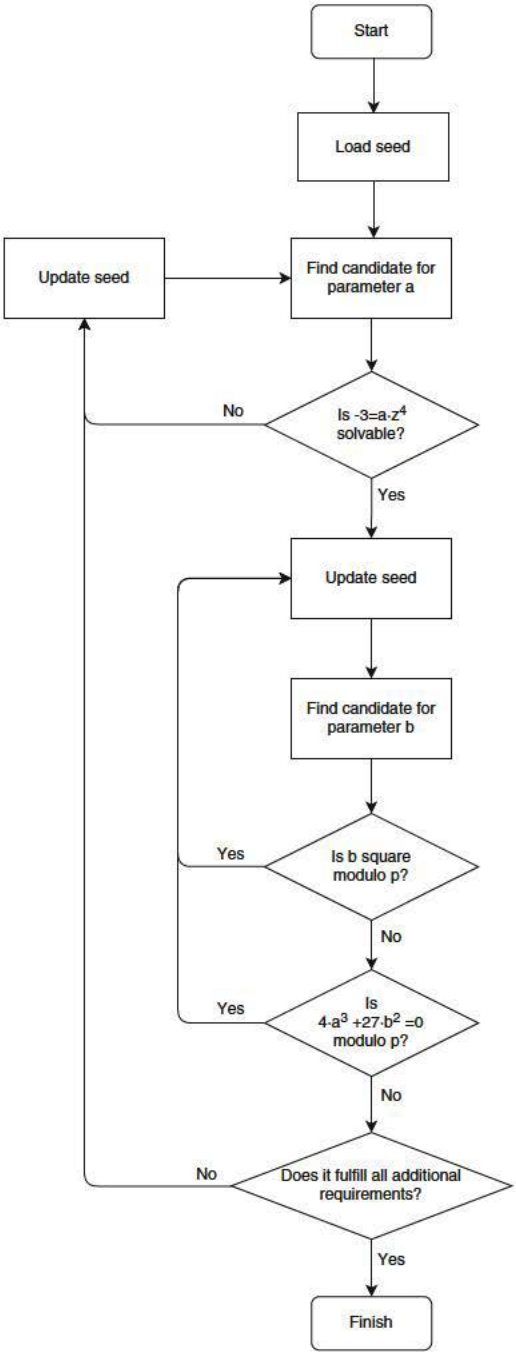

Fig. 16.2 Validation of candidates for parameters a and b in Brainpool

16.3.6 Factorizations

Two of the security requirements defined by Brainpool imply the factorization of integers. In one case, it is necessary to factorize the value $|E| - 1$, where q is the order of the elliptic curve, in order to avoid attacks using the Weil or Tate pairings. Those attacks allow the embedding of the cyclic subgroup of the elliptic curve into the group of units of a degree-l extension field of $\mathbb{F}_p$, where subexponential attacks on the DLP exist [10, p. 5].

In the other case, the specification requests to factorize the value d, which is the square-free factor of $4p - u^2$, where $u = |E| - p - 1$, so it can be checked that the class number of the maximal order of the endomorphism ring of the elliptic curve is larger than 10^7 [10, p. 5].

16.4 SafeCurves

Researchers Daniel Bernstein and Tanja Lange explain in their website SafeCurves [6] how all the standards that include recommended elliptic curves have addressed ECDLP security but not ECC security. The standards and official documents analyzed by SafeCurves are ANSI X9.62 [3], IEEE P1363 [21], SEC 2 [35], NIST FIPS 186 [30], ANSI X9.63 [2], Brainpool [10, 25], NSA Suite B [32], and ANSSI FRP256V1 [1].

Bernstein and Lange state that elliptic curves designed to be ECDLP secure may be attacked if they are not implemented properly, which would allow, for example, to produce incorrect results for some rare curve points, leak secret data when the input is not a curve point, or provide secret data through branch or cache timing attacks. The authors believe that secure implementation of the previously mentioned standard curves is theoretically possible but very hard. In order to avoid those implementation problems, the authors propose to use new curves which allow simple and secure implementations.

SafeCurves includes an evaluation of 20 curves taken from several sources (one SEC 2 curve, one ANSSI curve, two curves provided as examples of bad design, two Brainpool curves, three NIST curves, 5 Montgomery curves, and 6 Edwards curves) showing that some of them do not pass all their security requirements, which are divided into 3 main groups: curve parameters, ECDLP security, and ECC security.

16.4.1 Curve Parameters

In 2006, Bernstein stated that prime fields "have the virtue of minimizing the number of security concerns for elliptic-curve cryptography" [4], citing as two examples [13] and [17]. Later, he affirmed that "there is general agreement that

prime fields are the safe, conservative choice for ECC" [6]. Sharing that point of view, other standards and recommendations like Brainpool [25], NSA Suite B [32], or BSI TR-03111 [11] consider only prime fields for all type of applications.

While in all the Brainpool curves the order of the generator G equals the number of points of the curve (i.e., the cofactor is 1), the Montgomery and Edwards examples from SafeCurves have cofactors whose value is either 4 or 8. Small subgroups attacks, which could take advantage of curves with a cofactor greater than 1, are easily deactivated either by implementing the appropriate software checks or by the inherent characteristics of the Montgomery and Edwards curves described in [6].

Regarding the key length, the curves analyzed by SafeCurves have one of the following lengths: 221, 222, 251, 255, 382, 383, 414, 448, 511, and 521 bits.

16.4.2 ECDLP Security

Bernstein and Lange analyzed how resistant are the curves against some well-known attacks to the ECDLP. More specifically, they studied the following characteristics:

- *Rho method*: Bernstein and Lange stated that the rho method [34] breaks ECDLP using, on average, approximately $\sqrt{\Pi|G|/4}$ additions, where $|G|$ represents the order of the generator of the set of elliptic curve points used in the computations. SafeCurves requires curves such that that value is over 2^{100}.
- *Transfers*: In the authors' language, a "transfer" converts the ECDLP into the DLP using linear algebraic groups. Multiplicative transfers were introduced in 1993 and are often referred to as "the MOV attack" [19, 27]. Additive transfers were introduced in 1998 and are sometimes called "the Smart attack" [18, 20]. SafeCurves checks if the elliptic curve is safe against additive and multiplicative transfers by computing a value associated to the parameters of the curve and checking if it is higher than a certain threshold.
- *Complex-multiplication field discriminant*: The number of rational points on an elliptic curve over $\mathbb{F}_p$ is $p + 1 - t$, where t is the trace of the curve, while the order of the generator G is a prime divisor of that value [12]. If s^2 is the largest square dividing $t^2 - 4p$, then it can be affirmed that $(t^2 - 4p)/s^2$ is a square-free negative integer. If $(t^2 - 4p)/s^2 \equiv 1 (\text{mod } 4)$, then D is defined as $(t^2 - 4p)/s^2$; otherwise, D is $4(t^2 - 4p)/s^2$. SafeCurves requires the absolute value of this complex-multiplication field discriminant D to be larger than 2^{100}.
- *Rigidity*: SafeCurves checks the degree to which the elliptic curve generation process is explained (e.g., the choice for the seed values, the operations performed on them to derive the parameters, etc.).

16.4.3 ECC Security

In this aspect of their study, Bernstein and Lange analyzed the following list of features:

- *Ladders*: The authors consider that the most important computation in ECC is the single-scalar multiplication. Montgomery curves support a very simple scalar-multiplication method, the Montgomery ladder [29], which is simpler than the standard short Weierstrass scalar-multiplication methods. The Montgomery ladder uses a single standard addition–subtraction–doubling chain, always following a simple, highly efficient double-add pattern.

  SafeCurves requires curves to support simple and constant-time multiplica-tions, avoiding conflicts between efficiency and security. Both the Montgomery and Edwards curves included in the study satisfy this requirement.
- *Twist security*: If the original curve has $p + 1 - t$ points, then any nontrivial quadratic twist has in turn $p + 1 + t$ points. In a generic implementation, programmers have to be careful enough to include different checks (e.g., that the point sent by the other party effectively belongs to the elliptic curve, that the order of the point multiplied by the cofactor is not equal to $\mathcal{O}$, etc.) [8]. This requirement checks if the elliptic curve under analysis is protected against attacks derived from those situations without forcing the programmer to include specific code to handle those situations. In the scope of their contribution, the authors define such a curve as a "twist-secure" curve.
- *Completeness*: This requirement checks if the curve equations allow to use complete single-scalar and multi-scalar multiplications, in the sense that the same single point addition formula must return valid values in all the cases (i.e., when one of the input points is P and the other is $-P$, P, $\mathcal{O}$, or any other point) [5].
- *Indistinguishability from uniform random strings*: Standard representations of elliptic-curve points are easily distinguishable from uniform random strings. This poses a problem for many cryptographic protocols using elliptic curves (e.g., censorship-circumvention protocols, password-authenticated key-exchange protocols, etc.). One of the workarounds for this problem is for the protocol to bounce randomly between a curve and its twist, but according to the authors this is a complicated and error-prone approach [9].

  In 2013, a team of researchers led by Bernstein and Lange proposed a solution to this problem using bijective maps. Hence, this requirement checks the availability of those bijective maps for the curves under analysis.

16.4.4 Results

Using the previously described criteria, the authors of SafeCurves evaluated the aforementioned 20 curves. As it was expected given the content of the requirements, all the curves originally described using the short Weierstrass form do not satisfy

at least three of the four ECC security requirements. Apart from that, the SEC 2 curve does not pass the complex-multiplication field discriminant test, while the NIST and the ANSSI curves are considered to be manipulable. Regarding the two Brainpool curves considered in the comparison, they pass all the parameter and ECDLP security requirements.

16.5 Conclusions

Selecting a secure elliptic curve is one of the most important steps when using an ECC algorithm. The Brainpool specification has been analyzed by the scientific community since its first appearance in 2005 and includes clear instructions that allow any interested researcher to replicate the curve generation procedure. However, when it was designed there were requirements that were not taken into account and that are important when deploying an ECC solution.

Compared to the Edwards and Montgomery curves analyzed in SafeCurves, the Brainpool curves do not fulfill some security requirements, though it could be argued that a careful implementation of the curves should avoid the attacks related to those requirements.

In addition to that, it is also important to note that, using standard coordinates, short Weierstrass curves outperform Edwards and Montgomery curves when computing scalar-multiplication operations. As it is usually the case, having a higher level of security by default comes at a cost.

Most software libraries and hardware devices (e.g., smart cards) released during the last years support curves using the short Weierstrass form, so the installed base of those curves is quite large. The implementation pace of Edwards and Montgomery curves is still slow, but it will definitely increase in the new years as new ECC protocols and applications are made public.

Acknowledgements This work has been partly supported by Ministerio de Economía y Competitividad (Spain) under the project TIN2014-55325-C2-1-R (ProCriCiS), and by Comunidad de Madrid (Spain) under the project S2013/ICE-3095-CM (CIBERDINE), cofinanced with the European Union FEDER funds.

References

1. Agence Nationale de la Sécurité des Systèmes d'Information. (2011). Avis relatif aux paramètres de courbes elliptiques définis par l'Etat français. http://www.legifrance.gouv.fr/affichTexte.do?cidTexte=JORFTEXT000024668816.
2. American National Standards Institute. (2001). Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography. ANSI X9.63.
3. American National Standards Institute. (2005). Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). ANSI X9.62.

4. Bernstein, D. J., & Lange, T. (2007). Curve25519: New Diffie-Hellman speed records. In *Proceedings of the 9th International Conference on Theory and Practice in Public-Key Cryptography (PKC 2006)* (pp. 207–228).
5. Bernstein, D. J., & Lange, T. (2007). *Faster addition and doubling on elliptic curves* (pp. 29–50). Berlin/Heidelberg: Springer.
6. Bernstein, D. J., & Lange, T. (2014). SafeCurves. http://safecurves.cr.yp.to/.
7. Bernstein, D. J., & Lange, T. (2016). Explicit-Formulas Database. https://hyperelliptic.org/EFD/.
8. Bernstein, D. J., Birkner, P., Joye, M., Lange, T., & Peters, C. (2008). Twisted Edwards curves. Cryptology ePrint Archive, Report 2008/013. http://eprint.iacr.org/2008/013.
9. Bernstein, D. J., Hamburg, M., Krasnova, A., & Lange, T. (2013). Elligator: Elliptic-curve points indistinguishable from uniform random strings. In *Proceedings of the 2013 Conference on Computer & Communications Security* (pp. 967–980).
10. Brainpool. (2005). ECC Brainpool Standard Curves and Curve Generation. Version 1.0. http://www.ecc-brainpool.org/download/Domain-parameters.pdf.
11. Bundesamt für Sicherheit in der Informationstechnik. (2012). Elliptic Curve Cryptography. BSI TR-03111 version 2.0. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03111/BSI-TR-03111_pdf.pdf?__blob=publicationFile.
12. Cohen, H., & Frey, G. (2006). *Handbook of elliptic and hyperelliptic curve cryptography*. Boca Raton, FL: Chapman & Hall/CRC.
13. Diem, C. (2003). The GHS attack in odd characteristic. *Journal of the Ramanujan Mathematical Society*, *18*, 1–32.
14. Durán Díaz, R., Gayoso Martínez, V., Hernández Encinas, L., & Martín Muñoz, A. (2016). A study on the performance of secure elliptic curves for cryptographic purposes. In *Proceedings of the International Joint Conference SOCO'16-CISIS'16-ICEUTE'16* (pp. 658–667).
15. Edwards, H. M. (2007). A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, *44*, 393–422.
16. ElGamal, T. (1985). A public-key cryptosystem and a signature scheme based on discrete logarithm. *IEEE Transactions on Information Theory*, *31*, 469–472.
17. Frey, G. (1998). How to Disguise an Elliptic Curve (Weil Descent). http://www.cacr.math.uwaterloo.ca/conferences/1998/ecc98/slides.html.
18. Frey, G. (2001). Applications of arithmetical geometry to cryptographic constructions. In *Proceedings of the 5th International Conference on Finite Fields and Applications* (pp. 128–161). Heidelberg: Springer.
19. Frey, G., & Ruck, H. (1994). A remark concerning *m*-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, *62*, 865–874.
20. Gaudry, P., Hess, F., & Smart, N. P. (2002). Constructive and destructive facets of Weil descent on elliptic curves. *Journal of Cryptology*, *15*, 19–46.
21. Institute of Electrical and Electronics Engineers: Standard Specifications for Public Key Cryptography. IEEE 1363 (2000).
22. Institute of Electrical and Electronics Engineers: Standard Specifications for Public Key Cryptography - Amendment 1: Additional Techniques. IEEE 1363a (2004).
23. International Organization for Standardization/International Electrotechnical Commission: Information Technology-Security Techniques-Encryption Algorithms—Part 2: Asymmetric Ciphers. ISO/IEC 18033-2 (2006).
24. Koblitz, N. (1987). Elliptic curve cryptosytems. *Mathematics of Computation*, *48*(177), 203–209.
25. Lochter, M., & Merkle, J. (2010). Elliptic curve cryptography (ECC) Brainpool standard curves and curve generation. Request for Comments (RFC 5639), Internet Engineering Task Force.
26. Menezes, A. J. (1993). *Elliptic curve public key cryptosystems*. Boston, MA: Kluwer Academic Publishers.
27. Menezes, A., Okamoto, W., & Vanstone, S. (1993). Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, *39*, 1639–1646.

28. Miller, V. S. (1986). Use of elliptic curves in cryptography. In *Lecture Notes in Computer Science* (Vol. 218, pp. 417–426). Berlin: Springer.
29. Montgomery, P. L. (1987). Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, *48*, 243–264.
30. National Institute of Standards and Technology: Digital Signature Standard (DSS). NIST FIPS 186-4 (2009).
31. National Institute of Standards and Technology: Report on Post-quantum Cryptography (2016). http://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf.
32. National Security Agency: NSA Suite B Cryptography (2009). http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml.
33. National Security Agency: Commercial National Security Algorithm Suite (2015). https://www.iad.gov/iad/programs/iad-initiatives/cnsa-suite.cfm.
34. Pollard, J. (1978). Monte Carlo methods for index computation mod *p*. *Mathematics of Computation*, *32*, 918–924.
35. Standards for Efficient Cryptography Group: Recommended Elliptic Curve Domain Parameters. SECG SEC 2 version 2.0 (2010).