Binary Edwards Curves for Intrinsically Secure ECC Implementations for the IoT

Jacques J. A. Fournier, Antoine Loiseau, Jacques Fournier

Binary Edwards Curves for intrinsically secure ECC implementations for the IoT

Antoine Loiseau[1,2] and Jacques J.A. Fournier[3]
[1]*CEA-Tech, Gardanne, France*
[2]*Mines Saint Etienne, Gardanne, France*
[3]*Univ. Grenoble Alpes, CEA Leti, DSYS/LSOSP, F-38000 Grenoble*
{*antoine.loiseau, jacques.fournier*}*@cea.fr*

Keywords: IoT, Elliptic Curves Cryptography, Binary Edwards Curves.

Abstract: Even if recent advances in public key cryptography tend to focus on algorithms able to survive the post quantum era. At present, there is a urgent need to propose fast, low power and securely implemented cryptography to address the immediate security challenges of the IoT. In this document, we present a new set of Binary Edwards Curves which have been defined to achieve the highest security levels (up to 284-bit security level) and whose parameters have been defined to fit IoT devices embedding 32-bit general purpose processors. We optimized the choice of the point generator with the w-coordinate to save a multiplication in the addition and doubling formulae. We manage to compute one step of the Montgomery Ladder in 4 multiplications and 4 squares. On top of the performance benefits, cryptography over such curves have some intrinsic security properties against physical attacks.

1 Introduction

The rising amount of connectivity and number of connected devices (with the associated services) has given rise to the concept of the "Internet of Things". The technical challenges pertaining to the IoT are daunting (Stankovic, 2014). Among those challenges, there is the management of a massively huge network (hundreds of billions by the next decade) and the even bigger amount of associated data to manage, store and analyse, the heterogeneity of today's different systems that have to co-exist, the resilience of such systems, the openness of such an infrastructure and, last but not least, security & privacy (Fournier et al., 2014). Security and privacy have already been identified as vital issues of the IoT and as key to its adoption and deployment (Skarmeta and Moreno, 2014) (Rubens, 2014). The main difficulties with deploying security for the IoT come from the heterogeneity of this *"system of systems"*, lack of standard for security and inter-device communications, lack of off-the-shelf trusted IoT devices and tools (respecting all power, size and security constraints) and lack of appropriate business model ('low end' devices will potentially be communicating with 'high end' ones...).

One of the key aspects for such systems is that end-to-end security has to be enforced, which means that the end-nodes of an IoT infrastructure shall also embed security features and applications. Asymmetric cryptography is one of such security tools and given the size, power and performance constraints pertaining to IoT end-nodes (which are usually low-end devices with limited resources) Elliptic Curves Cryptography (ECC) is by far the best fit for such use cases and this despite the recently growing fear of quantum computers: since the latter are thought to be able one day to efficiently solve the Discrete Logarithm Problem (DLP), post-quantum cryptosystems, like lattice based cryptography or cryptography based on isogenies are currently being studied. However, these cryptosystems as they exist today, cannot be used in the constrained IoT nodes due to the large keys and long computing times. ECC is still, in our opinion, the best short and mid term answer to address the immediate security needs for the IoT.

1.1 Elliptic Curve Cryptography for the IoT

ECC was introduced by Koblitz and Miller in (Koblitz, 1987) (Miller, 1986) where the Elliptic Curves Discrete Logarithm Problem (ECDLP) is presented and its exploitation for doing cryptography

over elliptic curves is explained. The main standardization effort for ECC has been through the FIPS 186-4 (Information Technology Laboratory, 2013) by the NIST. Alternative approaches have been proposed like (Brainpool, 2005) which introduces Brainpool curves, and more recently the Edwards curve introduced in (Bernstein et al., 2011) by Bernstein *et al.* based on works by Edwards (Edwards, 2007). The latter pieces of work open a new vision in the landscape of applied elliptic curve cryptography.

The so-called "NIST curves" are largely deployed in several devices with different goals in today's IT systems ranging from low power devices to modern computers. However these curves were proposed in 1999 where "low power devices" were not as common as today and hence these curves were not designed for today's IoT constraints. Elliptic Curves usually used for cryptography are defined by the Weierstrass equation

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

and defined over a prime field with large prime characteristic or over binary extension fields of characteristic 2. The underlying mathematics in each kind of field have different arithmetics, each with its own advantages and drawbacks.

To start with, the arithmetics of the Weiesrtrass form are not adapted to the low resource constraints of IoT devices. Moreover those curves were neither designed nor chosen with the additional constraint of being resistant to side channel attacks and fault attacks: for example, due to the incomplete law group of these NIST curves, we have one formula for adding points and another formula for doubling points (except for the specific case Weierstrass curves over prime fields with $p > 3$ where we have a new complete formula for the addition (Renes et al., 2016)). Such constructions introduce in an ECC implementation weaknesses exploited in Simple Power Analysis (SPA) or Timing attacks (Fan et al., 2010a). In such cases, adding countermeasures against side channel attacks severely downgrade the performance and efficiency of such ECC implementations, which can be a heavy drawback for the IoT end-nodes. This is why it would be more interesting to look for curves that could avoid such weaknesses. For example curves like Curve25519, (Bernstein et al., 2011) (Edwards, 2007) have a complete group law providing one same formula for point addition and point doubling.

Another important aspect for an ECC implementation is the choice between curves over $\mathbb{F}_p$ or $\mathbb{F}_{2^m}$. Curves over large prime characteristics are usually preferred to those over binary fields due to claims that there may be sub-exponential algorithms for solving the discrete logarithm problem for elliptic curves on binary fields. However even if the best optimistic conjectures of the DLP claim to have a complexity of $L_{2^n}(\frac{2}{3})$[1] or even $L_{2^n}(\frac{1}{2})$, (Galbraith and Gaudry, 2016) recalls that they are still "hypotheses" and that recent experiments *"raise the doubt about the subexponentiality claims"*. One of the main advantages of arithmetics in binary fields is the carry-less operation because in practice managing carry propagations as needed in operations over prime fields downgrades computation times especially when it comes to implementing dedicated hardware accelerators. Moreover carry propagations can introduce weaknesses against side channel attacks (Fouque et al., 2008b).

We have a large set of models on large prime fields (Doche et al., 2006) (Edwards, 2007) (Castryck et al.,) (Bernstein et al., 2015) (Hisil et al., 2009) (Feng et al., 2010) (Moloney et al.,) (Bernstein et al., 2008a).

1.2 Our approach and contribution

In (Edwards, 2007), Harold Edwards introduced a very efficient model with a complete law group:

$$x^2 + y^2 = 1 + dx^2y^2$$

However, with the aforementioned drawbacks of working on curves in large prime fields, Bernstein *et al.* (Bernstein et al., 2008b) introduced the binary version of Edwards curves with also the advantage of a complete law and obviously without the carry propagation problems :

$$d_1(x+y) + d_2(x^2+y^2) = xy + xy(x+y) + x^2y^2$$

Such a Binary Edwards Curve is, given the constraints of IoT devices, one of the best choices for implementing ECC, resistant to physical attacks, in current and future embedded devices : they are fast, have a complete group law and the underlying carry-less arithmetics would require low power and compact dedicated hardware accelerators.

So far, only one such BEC has been proposed in the literature, which does not even offer what is the admitted reference of "128-bit security level". Hence our approach has been to generate other BECs, with higher security levels and with parameters tailored to fit the hardware constraints of IoT devices. To build those new elliptic curves, we split the parameters of the curve into three parts, each of which has been optimized separately. The first part is the finite field representation. The second is the curve's parameters which depend on the finite field representation and elliptic curve model chosen. The third part is the choice of the point generator of the elliptic curve group.

[1] $L_q(a) = e^{c\log(q)^a \log\log(q)^{1-a}}$

In this paper we provide the newly-generated BECs after explaining how they have been generated, i.e. by detailing the constraints and optimisations used in each of the three steps of the BECs generation. This new set of curves targets IoT implementations. Each curve was chosen to achieve from 112-bit to 284-bit security levels and the associated parameters have been chosen to be efficiently suited for hardware architectures working on data paths of 32 bits or 64 bits like those traditionally found in off-the-shelf IoT devices.

The methodology used for generating the curves and their associated parameters is depicted in Figure 1. We split security requirements in two sets. The first refers to criteria on the number of points of the curve and its twist, which is critical in the selection of a new elliptic curves. The second set of criteria contains all remaining criteria in 3.3. These criteria are checked after the construction of a optimized point generator.

The rest of this paper is organised as follows. Section 2 presents the concept of friendly polynomials from (Scott, 2007) and the associated advantages. We select a set of new binary fields to target different security levels to allow a good security/efficiency trade-off. Section 3 introduces the Binary Elliptic Curves as defined in (Bernstein et al., 2008b) with a focus on those curves having efficient arithmetics. We also review the different security requirements to select a BEC. Section 4 details how we built the generator of the elliptic curve subgroup used and how the choices made provide performance gains. Section 5 summarizes the expected intrinsic security properties of such BEC-based implementations against physical attacks. Some concluding remarks and perspectives are finally provided in the last section of this paper. Finally, the section 6 provides a first overview of performances of this kind of curves. The complete set of newly generated curves and parameters are given in the appendix.

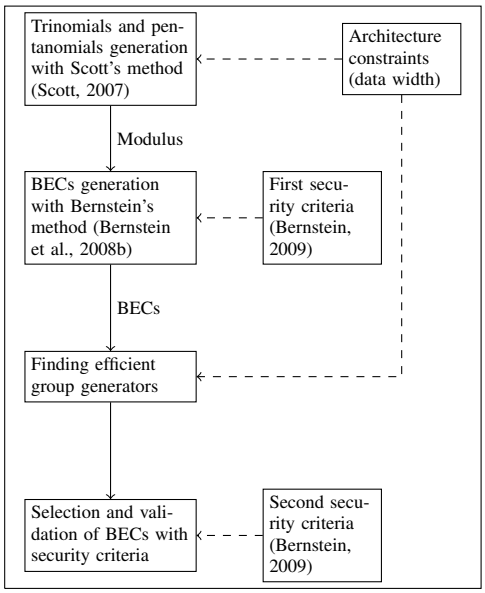

Figure 1: Overview of the generation scheme of BEC.

2 Optimal finite fields for efficient binary field arithmetics

The first step for generating a new set of elliptic curves for cryptography is to build "friendly" (i.e. for which calculations can be done rapidly) finite fields with an optimal arithmetic (i.e. for which some calculation "tricks" can be used for fast and compact implementations) for the targeted IoT devices. Arithmetics over elliptic curves depend on the representation of the base finite field which in turn depends on the choice of the modulus, represented under a polynomial form, used to define the finite field. If we want to consider the constraints due to the hardware platform used for the implementation (in our case, an IoT device embedding a 32-bit processor), choosing an irreducible polynomial at random, *i.e.* with a random number of monomials, seems the wrong approach if we want the best trade-off between performance and security. The main operation in such implementations is the reduction of a finite field element by the modulus and if we use a random polynomial, this operation could be expensive in terms of execution time and power consumption. A choice usually made for polynomials in binary fields $\mathbb{F}_{2^n}$ is a trinomial or a pentanomial of the type:

$$p(x) = x^n + x^a + 1, \qquad n > a > 0$$
$$p(x) = x^n + x^a + x^b + x^c + 1, \quad n > a > b > c > 0$$

The NIST (Information Technology Laboratory, 2013) for example selected its polynomials with the ANSI X9.62 rules also provided by Certicom (Johnson et al., 2001). A trinomial with the minimal coefficient a shall be selected if it exists. If no trinomial exists a pentanomial is chosen with minimal coefficients a, b and c.
However, as suggested in (Scott, 2007), the construction of the binary field is important for the efficiency of the implementation and this depends on the architecture used. Let w be the length in terms of bits of the registers of the targeted architecture, Scott (Scott, 2007) then defines three kinds of polynomials:

- Lucky Trinomial: A trinomial with $n - a \equiv 0 \mod w$.

- Lucky Pentanomial: A pentanomial with $n - a \equiv 0 \mod w$, $n - b \equiv 0 \mod w$ and $n - c \equiv 0 \mod w$.

- Fortunate Pentanomial: A pentanomial with at least two of these equations are verified : $n - a \equiv 0 \mod w$, $n - b \equiv 0 \mod w$ and $n - c \equiv 0 \mod w$.

A lucky trinomial is rare and a lucky pentanomial does not exit if $4|n$ or $n \equiv \pm 3 \mod 8$. These kinds of polynomials can save several shift operations compared to a classical polynomial used in cryptography. With a simple script in Sage (The Sage Developers, 2017) we searched all lucky trinomials and pentanomials for fields of bit lengths between 163 and 571: we found only 6 lucky trinomials compliant with a 32-bit architecture, i.e. whose representation is a multiple of 32. As for pentanomials fitted for 32-bit architectures, we have a larger number of them over several finite fields.

In the case of the field $\mathbb{F}_{2^{257}}$, we found the lucky trinomial $x^{257} + x^{65} + 1$, but on a 32-bit architecture (or any other architecture actually), a field element shall be stored onto 9 registers with only one bit in the last (uppermost) register. We might "waste" many store and load operations for only one bit. Such choices could have severe drawbacks for implementation of ECC in IoT devices. So, we select our own fields and polynomials to maximize the number of bits in the last register so as to minimize the number of load and store instructions used (even if we later keep in the end $\mathbb{F}_{2^{257}}$ as intermediate field between $\mathbb{F}_{2^{223}}$ and $\mathbb{F}_{2^{313}}$).

We first selected a set of lucky trinomials for 32 architectures with different sizes to target different security levels. We split our searches/findings into layers of minimum security levels of 112, 128, 192, 224 and 256 bits.

We also realised that most of those polynomials would also be compatible with 64-bit platforms (in case we wanted to anticipate for the day when such platforms would be available for IoT devices), except for the trinomials of $\mathbb{F}_{2^{479}}$ and $\mathbb{F}_{2^{521}}$. So we did the additional effort of looking for polynomials compatible with 64-bit data widths for the latter levels of security: hence we ended up with two additional pentanomials in $\mathbb{F}_{2^{487}}$ and $\mathbb{F}_{2^{569}}$.

The list of polynomials hence generated is as follows:

$$\mathbb{F}_{2^{223}} : x^{223} + x^{159} + 1$$
$$\mathbb{F}_{2^{257}} : x^{257} + x^{65} + 1$$
$$\mathbb{F}_{2^{313}} : x^{313} + x^{121} + 1$$
$$\mathbb{F}_{2^{431}} : x^{431} + x^{303} + x^{239} + x^{111} + 1$$
$$\mathbb{F}_{2^{479}} : x^{479} + x^{255} + 1$$
$$\mathbb{F}_{2^{487}} : x^{487} + x^{295} + x^{167} + x^{39} + 1$$
$$\mathbb{F}_{2^{521}} : x^{521} + x^{489} + 1$$
$$\mathbb{F}_{2^{569}} : x^{569} + x^{441} + x^{313} + x^{121} + 1$$

Table 1 compares these polynomials with those from the NIST in terms of number of logical operations (xor, shift, and) for a general purpose 32-bit architecture. We targeted a generic 32-bit architecture (excluding ARM-like architectures like where pre-shift operations are available to replace the sequence of logical instructions `XOR-AND`).

At this stage, we have selected a set of finite fields to match different security levels from 112 to 256 bits. With these selected binary fields we can now build the corresponding Binary Edwards Curves.

3 Binary Edwards Curves

Among the recently proposed models of elliptic curves for cryptography, Binary Edwards Curves (BEC) exhibit several interesting properties, (Bernstein et al., 2008b) (Bernstein, 2009). First, these curves favour implementations that will be intrinsically safe against a set of side channel and fault attacks (see Section 5). In fact, these curves have a complete group law and can be used with the Montgomery ladder algorithm. Moreover, the binary field arithmetics avoid leakages exploiting the carry propagation. Second, these curves allow an efficient arithmetic when using the correct coordinates' representations and fast arithmetics in binary fields. Thanks to the latter characteristics, BECs might, in our opinion, be interesting candidates for IoT applications.

3.1 Construction and definition

In (Edwards, 2007), Harold Edwards introduced a new model for elliptic curves allowing efficient computations for the elliptic curves' arithmetics. The main interesting feature for us is the complete group law on this curve giving one same formula for addition and doubling . The Edwards equation is:

$$x^2 + y^2 = 1 + dx^2y^2$$

Security level	Polynomials	# logical operations	# 32-bit registers
80	$x^{163}+x^7+x^6+x^3+1$	74	6
112	$x^{223}+x^{159}+1$	**38**	**7**
	$x^{233}+x^{74}+1$	62	8
128	$x^{257}+x^{65}+1$	**44**	**9**
	$x^{283}+x^{12}+x^7+x^5+1$	152	9
	$x^{313}+x^{121}+1$	**53**	**10**
192	$x^{409}+x^{87}+1$	108	13
	$x^{431}+x^{303}+x^{239}+x^{111}+1$	**99**	**14**
224	$x^{479}+x^{255}+1$	**78**	**15**
	$x^{487}+x^{295}+x^{167}+x^{39}+1$	**114**	**16**
256	$x^{521}+x^{489}+1$	**79**	**17**
	$x^{569}+x^{441}+x^{313}+x^{121}+1$	**135**	**18**
	$x^{571}+x^{10}+x^5+x^2+1$	296	18

Table 1: Comparison (in terms of number of logical operations) between different polynomials as modulus (the ones in bold characters are ours).

where d is not a square in $\mathbb{F}_p$. In (Bernstein et al., 2008b) (Bernstein, 2009), Bernstein *et al.* constructed an equivalent to Edwards curves onto binary fields.

Definition 1. *Binary Edwards Curves (BEC): Let K be a field of characteristic 2 and let $d_1, d_2 \in K$ with $d_1 \neq 0$ and $d_2 \neq d_1^2 + d_1$. The Binary Edwards Curve, with coefficients d_1 and d_2, is the affine equation:*

$$E_{d_1,d_2} : d_1(x+y)+d_2(x^2+y^2) = xy+xy(x+y)+x^2y^2$$

From this definition, we have the following interesting proprieties. This curve is symmetric and we have $\forall P \in E(K), P=(x,y) \Rightarrow -P=(y,x)$. With this assumption, we define the neutral point of the group law by $(0,0)$.

Definition 2. *Projective closure: The projective closure of the BEC E_{d_1,d_2} is:*

$$d_1(X+Y)Z^3+d_2(X^2+Y^2)Z^2 = XYZ^2+XY(X+Y)Z+X^2Y^2$$

The last important aspect of this construction is that each BEC is birationally equivalent to a Weierstrass form:

$$v^2+uv = u^3+(d_1^2+d_2)u^2+d_1^4(d_1^4+d_1^2+d_2^2)$$

We can hence define the mapping $\phi(x,y) \to (u,v)$ for this equivalence by:

$$u = d_1(d_1^2+d_1+d_2)\frac{x+y}{xy+d_1(x+y)}$$

$$v = d_1(d_1^2+d_1+d_2)\left(\frac{x}{xy+d_1(x+y)}+d_1+1\right)$$

And the inverse mapping $\phi^{-1}(u,v) \to (x,y)$ by:

$$x = \frac{d_1(u+d_1^2+d_1+d_2)}{u+v+(d_1^2+d_1)(d_1^2+d_1+d_2)}$$

$$y = \frac{d_1(u+d_1^2+d_1+d_2)}{v+(d_1^2+d_1)(d_1^2+d_1+d_2)}$$

3.2 Arithmetics on BEC

There are special cases where the arithmetics associated with the BEC can be optimised. For the case where $d_1 = d_2$ we have the following equation for a BEC:

$$d(x+y+x^2+y^2) = xy+xy(x+y)+x^2y^2 \qquad (1)$$

with the condition $\nexists t \in K \Rightarrow d = t^2+t$. In the following sections, we consider this particular case of BECs.

Addition and doubling: Efficient arithmetics on elliptic curves depend on the choice for the coordinates' representation. In the case of BECs, the projective w-coordinate with differential additions and doublings (Bernstein et al., 2008b) is the fastest way for doing points addition and point doubling. The idea of the differential w-coordinate is to represent a point $P(x,y)$ of the curve by $w(P) = x+y$. In this way, we can compute $w(2P)$ and $w(Q+P)$ given $w(P)$, $w(Q)$ and $w(Q-P)$. Let P and Q be two points of the curve, and let $w(P) = W_2/Z_2$, $w(Q) = W_3/Z_3$ and $w(Q-P) = w_1$, we compute $w(2P) = W_4/Z_4$ and $w(Q+P) = W_5/Z_5$ with Algorithm 1. This kind of coordinates is well suited for implementing the Montgomery ladder (see Algorithm 3) to compute the exponentiation of a point.

For an "Add and Double operation" we have only 5 field multiplications, 4 squares and 2 multiplications by d. The Montgomery ladder computes $w(kP)$ and $w(kP+P)$ with an addition and a doubling at each step. The use of w-coordinates appears to be the best way to compute scalar exponentiations. In the Algorithm 1 for the addition and doubling we use a scalar $w_1 = w(Q-P)$. In the case of the Montgomery ladder (Joye and Yen, 2002) as given in Algorithm

Algorithm 1 w-coordinates Adding and Doubling

Require: W_2, Z_2, W_3, Z_3, w_1
$C \leftarrow W_2.(Z_2 + W_2)$
$V \leftarrow C.W_3.(Z_3 + W_3)$
$W_4 \leftarrow C^2$
$Z_4 \leftarrow d(Z_2^2)^2 + W_4$
$Z_5 \leftarrow V + d(Z_2.Z_3)^2$
$W_5 \leftarrow V + Z_5.w_1$
return W_4, Z_4, W_5, Z_5

3, the difference $R_1 - R_0$ is invariant at each step of the ladder. In other words the scalar w_1 is a constant scalar.

Recently, new formulae for w-differential adding and doubling have been proposed to save a multiplication by d (Kim et al., 2014) (Koziel et al., 2015): an additional squaring can be saved with the Co-Z trick which performs $\frac{W_2}{Z} + \frac{W_3}{Z} = \frac{W_4}{Z'}$ and $2 \times \frac{W_2}{Z} = \frac{W_5}{Z'}$. Finally, the addition and doubling operations can be written as :

$$\frac{W_4}{Z'} = \frac{(W_2 + W_3)^2 + \frac{1}{w_1}(W_2 + W_3)^2}{Z^2 + \frac{1}{w_1}(W_2 + W_3)^2}$$

$$\frac{W_5}{Z'} = \frac{(W_2(W_2 + Z))^2}{dZ^4 + (W_2(W_2 + Z))^2}$$

Algorithm 2 gives the computation of these coordinates. In this case the Adding and Doubling operation takes 5 multiplications, 1 multiplication by d and 4 squarings.

Algorithm 2 w-coordinates Adding and Doubling revisited with the Co-Z trick.

Require: $W_2, W_3, Z, \frac{1}{w_1}$
$C \leftarrow (W_2 + W_3)^2$
$D \leftarrow Z^2$
$E \leftarrow \frac{1}{w_1} C$
$U \leftarrow E + C$
$V \leftarrow E + D$
$S \leftarrow (W_2(Z + W_2))^2$
$T \leftarrow S + dD^2$
$W_4 \leftarrow UT$
$W_5 \leftarrow VS$
$Z' \leftarrow VT$
return W_4, W_5, Z'

With this kind of coordinates (with the Co-Z trick), we loose the completeness of the addition group law in favour of an "almost" complete addition law. In fact, if $w(P-Q) = 0$ we cannot define $\frac{1}{w_1}$ used in the Algorithm 2. Recently, (Rezaeian Farashahi and Hosseini, 2017) defined a new kind of coordinates also "almost" complete with the same efficiency

Algorithm 3 Montgomery Ladder

Require: $w(P), k = (k_{t-1}, ..., k_0)_2$
$R_0 \leftarrow O$
$R_1 \leftarrow P$
for $j = t - 1$ **to** 0 **do**
  if $k_j = 0$ **then**
    $R_1 \leftarrow R_0 + R_1$
    $R_0 \leftarrow 2R_0$
  else
    $R_0 \leftarrow R_0 + R_1$
    $R_1 \leftarrow 2R_1$
  end if
end for
return $R_0 = w(kP), R_1 = w(kP + P)$

as the mixed differential w-coordinates with Co-Z trick. However, (Rezaeian Farashahi and Hosseini, 2017) uses an advanced definition of w by $w(x,y) = \frac{x+y}{d(x+y+1)}$ and does not use the Co-Z trick. In this way, to recover affine coordinates from the Montgomery Ladder's output we have to compute two inversions against only one for mixed differential w-coordinates with Co-Z trick.

Recovering the affine coordinates from the w-coordinate: The Montgomery Ladder (Algorithm 3) computes $w_2 = w(kP)$ and $w_3 = w(kP + P)$. We can calculate the x coordinate of kP with the output of the Montgomery Ladder and with $P = (x_1, y_1)$ using the following equation from (Bernstein et al., 2008b):

$$x^2 + x = \frac{w_3(d + w_1 w_2(1 + w_1 + w_2) + w_1^2 w_2^2) + d(w_1 + w_2) + (y_1^2 + y_1)(w_1^2 + w_2)}{w_1^2 + w_1}$$

This formula requires 1 inversion, 4 multiplications and 4 squares. From there we have to solve a equation $x^2 + x = A$ if the trace of A is zero. For a fixed generator point, we can save the inversion operation if $w_1^2 + w_1$ is precomputed. We can also recover the y coordinate with

$$y^2 + y = \frac{d(x + x^2)}{d + x + x^2}$$

3.3 Generating secure Binary Edwards Curves

The first issue we wanted to resolve was to have curves that provide the required level of security (i.e. at minimum at 128-bit level of security as recommended by the NIST since December 2015). So far, to the best of our knowledge, the only BEC curve proposed by Bernstein in (Bernstein, 2009) has a security level of 125 bits.

The first things we started implementing were methods and tools to generate and propose a set of BECs with different field sizes for different security

levels in order to match a targeted device's requirements and the best security/efficiency (power and performance) trade-offs.
In (Bernstein, 2009) the authors enumerate the following security requirements for a BEC (and ECC in general):

- *Prime degree of the finite field extension:* The choice of the binary field extension is essential for fast computing but an extension field with a non-prime degree can be attacked by the GHS algorithm (Gaudry et al., 2002). Moreover a large prime degree has to be chosen due to the birthday paradox and has to provide a sufficient level security. We shall target a minimum security level of 128 bits. In this way we shall use fields with an extension degree of at least 256 bits.
- *Number of points:* The curve E shall have a cardinal of $2^c p$ where p is a large prime and c shall be small (1, 2 or 3).
- *Number of points on the Twist:* We have the same requirement on the number of points of the twist of the curve E in order to ensure resistance against some fault attacks on the original curve.
- *j-invariant:* The j-invariant of the BEC, $1/d^8$, shall generate the extension field.
- *Avoiding small discriminants:* The discriminant $\Delta_E = \mathrm{Tr}(E)^2 - 4q$ shall be divisible by a large prime exactly once.
- *Avoiding pairing attack:* The multiplicative order k of 2^m modulo the large prime of $|E|$ shall be large. We have the same condition with the prime of the twist. k is called the embedding degree of the subgroup of the elliptic curve. NIST advises in (Information Technology Laboratory, 2013) to have an embedding degree greater than 20. However with the recent improvements of index calculus attacks (Galbraith and Gaudry, 2016), we have to increase this requirement. Brainpool curves (Brainpool, 2005) have an embedding degree greater than $\frac{l-1}{100}$, where l is the prime order of big subgroup of the elliptic curve. This requirement is somehow "overrated", but we choose to check it anyway, as it is very easy for an elliptic curve with a large subgroup of prime order to satisfy this criterion.

Based on these requirements, we can check, for each d, if a given curve is secure. In practice, we start with parameters d with a small Hamming Weight, first with binomials, trinomials, quadrinomials and finally with pentanomials. Almost all tested curves fail to the "number of points" criterion. Hence the principal operation for generating & testing a BEC is to run a point counting algorithm. We implemented the AGM method (Cohen et al., 2006) to perform this counting. Moreover, to avoid counting the number of points on the twist, we use the relationship between a curve E and its twist E^{tw} such that $|E| + |E^{tw}| = 2q + 2$ to determine the number of points on the twist.

Using the above implemented algorithms, we managed to build new BECs on alternative fields with optimal finite field arithmetics and respecting all the security requirements. We implemented this algorithm in C language and based on the FLINT library (FLINT, 2015). For our first trial, the program was run over an Intel i5-6200 at 2.40GHz over a Linux virtual machine on Windows. After two full days of running, we built the following BEC on $\mathbb{F}_{2^{313}}$:

$$\mathbb{F}_{2^{313}}[t] = \frac{\mathbb{F}_2[X]}{(X^{313} + X^{121} + 1)}$$
$$d = t^{38} + t^{33} + t^{28} + 1$$

$$|E| = 2^2 \times (2^{311} - 2380103563917833578062289719146201818615624174 5)$$
$$|E^{tw}| = 2 \times (2^{312} + 4760207127835667156124579438292403637231248349 1)$$

This curve verifies all the above security requirements as illustrated below:

- 313 is a prime number and avoids the GHS attack on $\mathbb{F}_{2^{313}}$.
- The number of points of E in $\mathbb{F}_{313}$ is $4p_1$ where p_1 is a large prime.
- The number of points of the quadratic twist of E, E^{tw}, is $2p_2$ where p_2 is a large prime.
- The j-invariant $1/d^8$ generates $\mathbb{F}_{2^{313}}$.
- $(2^{313} + 1 - 4p_1)^2 - 2^{315}$ is divisible by a large prime and avoids small discriminants.

$$\frac{(2^{313} + 1 - 4p_1)^2 - 2^{315}}{-1887935404880658993146706844859746511}$$

- The multiplicative order of 2^{313} modulo p_1 is not small, exactly $(p_1 - 1)/2$. The multiplicative order of 2^{313} modulo p_2 is not small, exactly $p_2 - 1$.

Likewise other curves with different sizes have been generated as given in Appendix.

4 Optimal group generator

Now that we have set methods and tools for optimal (i.e. respecting our security and 32-bit oriented implementation constraints) field representations and parameter d for a BEC, we have now to choose a generator of the elliptic curve sub-group used to

perform an exponentiation. In this case, the security of the protocol is based on the difficulty of the DLP in this sub-group. In fact DLP is hard to solve onto elliptic curves group. We selected a curve with a large sub-group, *i.e.* with a number of points divisible by a large prime. In our case we have curves with a number of points $n = 2^c p$ with $c = 1, 2$ or 3 and p a large prime. So, we look for a point of the curve of order p.

We can select a random point G of order p but, in this case, the w-differential coordinate of G is a random scalar. However in the "Add and Double" formula of the BEC (Algorithm 2) we have a multiplication by the scalar $\frac{1}{w_1}$. Due to the property of the Montgomery Ladder, $\frac{1}{w_1}$ is invariant at each step of the ladder. In this case, if we can find a generator G with a $\frac{1}{w(G)}$ having a small Hamming Weight, we can expect to reduce the number of multiplications for the addition and doubling steps from 5 to 4 multiplications.

If we consider $w = x + y$, then from the equation 1 of a BEC with a parameter d, we have a new equation of BEC:

$$d(w+w^2) = x^4 + (1+w+w^2)x^2 + (w+w^2)x \quad (2)$$

So, for a chosen w we can find an x coordinate from the equation 2 and a y coordinate from the initial equation 1. We choose an inverse w-coordinate for G with a small Hamming Weight.

In the previous section we took as example the BEC over $\mathbb{F}_{2^{313}}$ with modulus $x^{313} + x^{121} + 1$ and $d = t^{38} + t^{33} + t^{28} + 1$. If we use this method to build a friendly generator for this curve we find a generator G with:

G_x =0x15c67e3024c7c27466e72a3391256e9a729fc15
8092053d89087c0f38408b214b0ade57363ea938

G_y =0x15c67e3024c7c27446e72a3391256e9a529fc15
8092053d8b087c0f38408b214b0ade57363ea938

$G_{1/w}$ = 0x10000000000000001

$\frac{1}{w}$ is reduced to $t^{64} + 1$. So, the multiplication by w_1 in the "Add and Double" formula, Algorithm 2, is simply done using 9 XOR operations.

5 Considering physical attacks

We are working on a software implementation of these new curves for a 32 bit RISC V architecture. The implementation has been done keeping in mind resistance to side channel and fault attacks, based on several factors: the intrinsic choices made during the curves' and parameters' generations, the algorithms used for implementing some of the basic operations and the additional countermeasures from the literature. For example the completeness proprieties of BECs and the use of the Montgomery ladder offer a first level of security against side channel attacks. Moreover we have hard-coded the d parameter of the curve and the $\frac{1}{w}$ parameter of the point generator to avoid any attack of invalid curve analysis and invalid point analysis. Against attacks like Correlation Power Analysis (CPA), Template Attacks, Horizontal Attacks or Differential Fault Attacks (DFA) we added countermeasures like randomization of the coordinates and of the scalar (Fan et al., 2010a). Table 2 summarizes the theoretical advantages of the BEC model with respect to Side Channel Attacks (SCA) and Fault Attacks (FA).

6 Performances

We implemented this new set of elliptic curves on a RISC V core running at 100 MHz. No special instruction has been added to the RISC V, which means that we have a software only implementation. We have decided to implement the differential w-coordinates system, as described above, with two cases, one with the optimized generator and the other with a random generator. We added the projective coordinates system for an interesting comparison and to show how the w-coordinates could be useful for ECC implementations. For the finite field multiplication in $GF(2^n)$ we use a López-Dahab multiplication (Aranha et al., 2010). The exponentiation computation is performed by using a Powering Montgomery Ladder (Joye and Yen, 2002). All functions for the finite field arithmetics are written in Assembly Language and functions for ECC arithmetics in C language. The performances are given in Table 3 and the parameters of these new set of BECs are provided in the appendix. These performances are for a computation of kP without countermeasures, even if as explained in Section 5 and in Table 2, we are theoretically secure against against a set of side channel attacks and fault attacks. First of all, the gap between projective coordinates and w-coordinates is huge, of the order of 355% between both coordinates system. The gain in performance between an optimized generator and a random generator in w-coordinate is around 19%. In terms of absolute time, this gain is low for small curves but for large curves this gain could be very important, almost 100ms for the largest curve. In

Physical Attacks	Intrinsic Resistance		Remaining vulnerability
	Due to choice of parameters of BEC	**Due to implementation done**	**Additional countermeasure implemented**
Timming Attack (Kocher, 1996)	Unified arithmetics	Use of Montgomery ladder	-
SPA (Fan et al., 2010b)		Constant time programming	
DPA (Kocher et al., 1999)	-	-	Randomization
Template Attack (Herbst and Medwed, 2008)			of coordinates (Coron, 1999)
Relative Doubling Attack (Yen et al., 2005)	-	-	Blinded scalar (Coron, 1999)
RPA/ZPA (Akishita and Takagi, 2003)	w-coordinates arithmetics	Direct implementation of the generator	-
Carry-based Attack (Fouque et al., 2008b)	Binary curves chosen	-	-
Horizontal Attack (Clavier et al., 2010)	-	-	Attack model to be tested
Safe error (Fan et al., 2010b)	-	Use of Montgomery ladder	-
Invalid point analysis (Biehl et al., 2000)	-	-	Verify that point in on the curve
Invalid curve analysis (Ciet and Joye, 2005)	Curves' parameters on Twist for eg	Direct implementation of curve parameter	-
Twist Attack (Fouque et al., 2008a)	Curves' parameter for Twist	-	-
Differential Fault Attack (Biehl et al., 2000)	-	-	Blinded scalar (Coron, 1999)

Table 2: SCA and Fault Attacks against BEC-based ECC

our opinion, the next step for better performance is to add a carry-less multiplication instruction to the set of the RISC V instructions. In fact, the finite field multiplication is the bottleneck for the computation of kP. It is hard to compare these results with other results in the literature. To the best of our knowledge, the performances presented in this paper are the first public results for a RISC V architecture. We can compare with performances of MbedTLS on an ARM M3 running at 96MHZ presented in (Tschofenig et al., 2015). For a security level of 128 bits, an ECDSA signature (where the main operation is the kP operation) takes 122 ms with the standard secp256r1 curve, which is to be compared to 46 ms (for the kP operation) in our case. Note moreover that the performances in (Tschofenig et al., 2015) are given for a sliding window implementation, which is very efficient in terms of timing performances, but still vulnerable to side channel and fault attacks.

7 Conclusion

In this paper, we described how we generated a new set of Binary Edwards Curves which might be efficient, secure and low power alternatives for implementing elliptic curves cryptography in IoT devices. The optimization of the w-coordinate of the point generator allows a speed up if these coordinates are used with the Montgomery Ladder. This new set of curves covers different security levels to offer the possibility of making the best trade-off between security and execution time. These curves are built with a 32-bit architecture in mind but they are also compliant with smaller architectures (8 and 16 bits) and larger ones (64 bits). Using Binary Edwards Curves might allow to have some intrinsic protections against a set of side channel and fault attacks. Future work will consist in validating the expected "physical security" properties in practice. As the BEC model allows a complete compliance with the Weierstrass model, we shall also study how the use of BEC curves can be compliant, in practice, with already deployed implementations of ECDSA on Weierstrass curves. We shall also work on the implementation of a dedicated hardware accelerator in order to take advantage of the carry-less operations.

REFERENCES

Akishita, T. and Takagi, T. (2003). Zero-value point attacks on elliptic curve cryptosystem. In *International Conference on Information Security*, pages 218–233. Springer.

Aranha, D., Dahab, R., López, J., and Oliveira, L. (2010).

Curves	w-coor., optimal generator	w-coor., random generator	proj. coor., random generator
BEC223	32 ms	39 ms	137 ms
BEC257	46 ms	57 ms	201 ms
BEC313	79 ms	96 ms	342 ms
BEC431	188 ms	231 ms	821 ms
BEC479	242 ms	299 ms	1064 ms
BEC487	264 ms	326 ms	1156 ms
BEC521	316 ms	390 ms	1388 ms
BEC569	396 ms	489 ms	1744 ms

Table 3: Performances of a kP operation without countermeasures over new BECs on a RISC-V processor at 100 MHz.

Efficient implementation of elliptic curve cryptography in wireless sensors. 4(2):169–187.

Bernstein, D. (2009). Batch binary Edwards. In *Lecture Notes in Computer Science*, volume 5677 LNCS, pages 317–336.

Bernstein, D., Birkner, P., Joye, M., Lange, T., and Peters, C. (2008a). Twisted Edwards curves. *Lecture Notes in Computer Science*, 5023 LNCS:389–405.

Bernstein, D., Chuengsatiansup, C., Kohel, D., and Lange, T. (2015). Twisted hessian curves. *Lecture Notes in Computer Science*, 9230:269–294.

Bernstein, D., Duif, N., Lange, T., Schwabe, P., and Yang, B.-Y. (2011). High-speed high-security signatures. *Lecture Notes in Computer Science*, 6917 LNCS:124–142.

Bernstein, D., Lange, T., and Rezaeian Farashahi, R. (2008b). Binary edwards curves. In *Lecture Notes in Computer Science*, volume 5154 LNCS, pages 244–265.

Biehl, I., Meyer, B., and Mller, V. (2000). Differential fault attacks on elliptic curve cryptosystems. In *Annual International Cryptology Conference*, pages 131–146. Springer.

Brainpool (2005). ECC Brainpool Standard Curves and Curves Generation v1.0. http://www.ecc-brainpool.org/.

Castryck, W., Galbraith, S., and Rezaeian Farashashi, R. Efficient arithmetic on elliptic curves using a mixed Edwards-Montgomery representation. *IACR Cryptology ePrint Archive*.

Ciet, M. and Joye, M. (2005). Elliptic curve cryptosystems in the presence of permanent and transient faults. *Designs, codes and cryptography*, 36(1):33–43.

Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., and Verneuil, V. (2010). Horizontal correlation analysis on exponentiation. In *International Conference on Information and Communications Security*, pages 46–61. Springer.

Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., and Vercauteren, F. (2006). *Handbook of elliptic and hyperelliptic curve cryptography*. Discrete mathematics and its applications. Chapman & Hall/CRC.

Coron, J.-S. (1999). Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In *Cryptographique Hardware and Embedded Systems*, volume 1717, pages 292–302. LNCS.

Doche, C., Icart, T., and Kohel, D. (2006). Efficient scalar multiplication by isogeny decompositions. *Lecture Notes in Computer Science*, 3958 LNCS:191–206.

Edwards, H. (2007). A normal form for elliptic curves. In *Bulletin of the American Mathematical Society*, pages 393–422.

Fan, J., Guo, X., De Mulder, E., Schaumont, P., Preneel, B., and Verbauwhede, I. (2010a). State-of-the-art of secure ECC implementations: A survey on known side-channel attacks and countermeasures. In *Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2010*, pages 76–87.

Fan, J., Guo, X., Mulder, E. D., Schaumont, P., Preneel, B., and Verbauwhede, I. (2010b). State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures. In *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 76–87.

Feng, R., Nie, M., and Wu, H. (2010). Twisted Jacobi intersections curves. *Lecture Notes in Computer Science*, 6108 LNCS:199–210.

FLINT (2015). FLINT: Fast Library for Number Theory. http://flintlib.org/.

Fouque, P.-A., Lercier, R., Ral, D., and Valette, F. (2008a). Fault attack on elliptic curve Montgomery ladder implementation. In *Fault Diagnosis and Tolerance in Cryptography, 2008. FDTC'08. 5th Workshop on*, pages 92–98. IEEE.

Fouque, P.-A., Réal, D., Valette, F., and Drissi, M. (2008b). *The Carry Leakage on the Randomized Exponent Countermeasure*, pages 198–213. Springer Berlin Heidelberg, Berlin, Heidelberg.

Fournier, J., Tria, A., Pebay, F., and Noguet, D. (2014). Security Challenges facing the IoT. In *Pan European Networks Science & Technology*, pages 66–67.

Galbraith, S. and Gaudry, P. (2016). Recent progress on the elliptic curve discrete logarithm problem. *Designs, Codes, and Cryptography*, 78(1):51–72.

Gaudry, P., Hess, F., and Smart, N. (2002). Constructive and destructive facets of weil descent on elliptic curves. *Journal of Cryptology*, 15(1):19–46.

Herbst, C. and Medwed, M. (2008). Using templates to attack masked montgomery ladder implementations of modular exponentiation. In *International Workshop on Information Security Applications*, pages 1–13. Springer.

Hisil, H., Wong, K.-H., Carter, G., and Dawson, E. (2009). Jacobi quartic curves revisited. *Lecture Notes in Computer Science*, 5594 LNCS:452–468.

Information Technology Laboratory (2013). Digital Signature Standard (DSS). Technical Report NIST FIPS 186-4, National Institute of Standards and Technology.

Johnson, D., Menezes, A., and Vanstone, S. (2001). The Elliptic Curve Digital Signature Algorithm (ECDSA). Technical report, Certicom.

Joye, M. and Yen, S.-M. (2002). The Montgomery powering ladder. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 291–302. Springer.

Kim, K., Lee, C., Negre, C., and Negre, C. (2014). Binary edwards curves revisited. *Lecture Notes in Computer Science*, 8885:393–408.

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):243–264.

Kocher, P. C. (1996). Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proceedings of Advances in Cryptology (CRYPTO'96)*, LNCS, pages 104–113. Springer-Verlag.

Kocher, P. C., Jaffe, J., and Jun, B. (1999). Differential Power Analysis. In *Proceedings of the 19th International Advances in Cryptology Conference (CRYPTO'99)*, number 1666 in LNCS, pages 388–397. Springer-Verlag.

Koziel, B., Azarderakhsh, R., and Mozaffari-Kermani, M. (2015). Low-resource and fast binary edwards curves cryptography. *Lecture Notes in Computer Science*, 9462:347–369.

Miller, V. (1986). Use of Elliptic-Curves in Cryptography. *Lecture Notes in Computer Science*, 218:417–426.

Moloney, R., McGuire, G., and Markowitz, M. Elliptic Curves in Montgomery Form with $B = 1$ and Their Low Order Torsion. *IACR Cryptology ePrint Archive*.

Renes, J., Costello, C., and Batina, L. (2016). Complete addition formulas for prime order elliptic curves. *Lecture Notes in Computer Science*, 9665:403–428.

Rezaeian Farashahi, R. and Hosseini, S. (2017). Differential addition on binary elliptic curves. *Lecture Notes in Computer Science*, 10064 LNCS:21–35.

Rubens, P. (2014). Internet of Things a Potential Security Disaster.

Scott, M. (2007). Optimal Irreducible Polynomials for $GF(2^m)$ Arithmetic. In *IACR Cryptology ePrint Archive*, volume 2007, page 192.

Skarmeta, A. and Moreno, M. (2014). Internet of Things. In Jonker, W. and Petkovic, M., editors, *Secure Data Management*, Lecture Notes in Computer Science, pages 48–53. Springer International Publishing.

Stankovic, J. (2014). Research directions for the internet of things. *IEEE Internet of Things Journal*, 1(1):3–9.

The Sage Developers (2017). *SageMath, the Sage Mathematics Software System (Version x.y.z)*. `http://www.sagemath.org`.

Tschofenig, H., Pegourie-Gonnard, M., and Vincent, H. (2015). Performance of State-of-the-Art Cryptography on ARM-based Microprocessors. In *NIST Lightweight Cryptography Workshop 2015 Session VII: Implementations & Performane*.

Yen, S.-M., Ko, L.-C., Moon, S., and Ha, J. (2005). Relative doubling attack against montgomery ladder. In *International Conference on Information Security and Cryptology*, pages 117–128. Springer.

APPENDIX

In this annex we propose a new set of binary elliptic curves with the finite field representation, BEC parameter and the Weierstrass form, a "friendly" (i.e. adapted to 32-bit architectures) generator G and coordinates in Weierstrass form. Each curve was generated using the previously described methods and they are composed of the following parameters:

- m is the size of the extension of $\mathbb{F}_2$.
- f is the modulus defining the representation of the finite field.
- d is the parameter of the curve in BEC form, equation 1.
- G_x, G_y and $G_{1/w}$ are the coordinates of a friendly generator in BEC form with $G_{1/w}$ the w-coordinate such as $G_{1/w} = \frac{1}{w}$ where $w = G_x + G_y$.
- n is the number of points of the curve.
- p is the prime order of the sub-group generated by G
- h is the cofactor.

We make the choice of generating a curve over 223 bits which does not reach the 128 bits security level due to the very interesting performance properties of such curves: in a very constrained environment on a very low power devices, a trade off between security and efficiency could be done.

$m =$ 223
$f =$ $x^{223} + x^{159} + 1$
$d =$ $t^{64} + t^{36} + t^5 + 1$
$G_x =$ 205bfedd71b0b0fdfeb3345af71cc721790e83 c4b88094e9a63f6d43
$G_y =$ 205bfeddf1b0b0fd7eb3345af71cc721790e83 c4b88094e9a63f6d43
$G_{1/w} =$ 100000001
$n =$ 13479973333575319897333507543509817 898877363662863184062623242975476
$p =$ 33699933333938299743333768858774544 7471934091571579601565581074386 9
$h =$ 4

$m =$ 257
$f =$ $x^{257}+x^{65}+1$
$d =$ $t^{65}+t^{31}+t^{14}+1$
$G_x =$ 16b46e24aa4b12ab2289fcd3417615387810f0
83f43419d8cae38ad9ac640d960
$G_y =$ 16b46e24aa4b12aba289fcd3417615383810f0
83f43419d8eae38ad9ac640d968
$G_{1/w} =$ 10000000000000000000000000000000000000
000000000000
$n =$ 231584178474632390847141970017375815706
332616967362709021140632923291797618908
$p =$ 578960446186580977117854925043439539265
831542418406772552851582308229494047 27
$h =$ 4

$m =$ 313
$f =$ $x^{313}+x^{121}+1$
$d =$ $t^{38}+t^{33}+t^{28}+1$
$G_x =$ 15c67e3024c7c27466e72a3391256e9a729fc
158092053d89087c0f38408b214b0ade5736
3ea938
$G_y =$ 15c67e3024c7c27446e72a3391256e9a529fc
158092053d8b087c0f38408b214b0ade5736
3ea938
$G_{1/w} =$ 10000000000000001
$n =$ 16687398718132110018711107079449625 8
95333629080816145622654549217988600 0
188954062243097663372 12
$p =$ 41718496795330275046777767698624064 7
38334072702040364056636373044971500 0
472385155607744158430 3
$h =$ 4

$m =$ 431
$f =$ $x^{431}+x^{303}+x^{239}+x^{111}+1$
$d =$ $t^{83}+t^{66}+t^{17}+1$
$G_x =$ 4e1765c1f2f6140db17d5ef2f14c59a38a93e5
b65ba9acca547bf2cc34f3d55bd85ccf4daeaf
7ca1becaa8ee877b01f8d8acae12b210
$G_y =$ 4e1765c1f2f6140d317d5ef2f14c59a30a93e5
b65ba9acca547bf2cc34f3d55b585ccf4daeaf
7ca13ecaa8ee877b01f8d8acae12b210
$G_{1/w} =$ 10000000000000001
$n =$ 55453393882416297191568283682861674 0
68728741507516331503409591613118822 2
85362041720505164102585754980032500 3
969881916422248862069 2
$p =$ 13863348470604074297892070920715418 5
17182185376879082875852397903279705 5
71340510430126291025646438745008125 0
992470479 1055622155173
$h =$ 4

$m =$ 479
$f =$ $x^{479}+x^{255}+1$
$d =$ $t^{73}+t^{29}+t^{3}+1$
$G_x =$ 7bdd9f19e11e888e80d7c093092d208b4fe99
6e8fcbdffa28cc90173ece2c43673f1372e975
ba9dcd3a06332abf15dbe9b679f6c63e30b88
4ab93272
$G_y =$ 3bdd9f19a11e888e40d7c093492d208b8fe99
6e8bcbdffa24cc90173ece2c436f3f1372e175
ba9dc53a063322bf15dbe1b679f6ce3e30b88
cab93272
$G_{1/w} =$ 10000000000000001
$n =$ 15608742751579961156907986148965831
528742990713324855754295784798126 5
869415448019717954458818867630469 4
698032411395977889616430979594599 5
58356
$p =$ 39021856878949902892269965372414578 8
21857476783312139385739461995317146 7
35386200492948861470471690761733674 5
081028489944724041077448986498639589
$h =$ 4

$m =$ 487
$f =$ $x^{487}+x^{295}+x^{167}+x^{39}+1$
$d =$ $t^{69}+t^{33}+t^{15}+1$
$G_x =$ 339b843c53c409543f396d39e57efde813f06
e3099735004b999b15776a75a4c3a22dcaf1e
91e261fe479b89a64d65103928195d727bd3
d157735b2071
$G_y =$ 339b843c53c40954bf396d39e57efde893f0
6e30997350043999b15776a75a4c3a22dcaf
1e91e261fe479b89a64d6510b928195d727b
d3d1d7735b2071
$G_{1/w} =$ 10000000000000001
$n =$ 39958381444044700561684444541352528 1
35820562261116307309972090832047582 0
18186213822669385452154446682534549 6
75533707766156971976964541332597784 4
$p =$ 99895953610111751404211111353381321 8
39551405652790768274930227080118956 0
45465534556673463630386116706336374 6
88834269415392429942411353331494461
$h =$ 4

$m =$ 521
$f =$ $x^{521}+x^{489}+1$
$d =$ $t^{66}+t^{29}+t^{28}+1$
$G_x =$ 16b369b497b805e6199a342909aa4608cdc
ecb10e0988ba73eb1f1186039c8b1f6d2a9
db39b1302d29d9d449b9aa459cc5d6bbb4e
33a1eb8fcc056ce724cde5aaa8
$G_y =$ 16b369b4b7b805e6199a342909aa4608cdc
ecb10e0988ba73eb1f1186039c8b1f6d2a9
db39b1302d29d9d449b9aa459cc5d6bbb4e
33a1eb8fcc056ce724cde5aaa8
$G_{1/w} =$ 100000001
$n =$ 6864797660130609714981900799081393 21
726943530014330540939446345918554318
339765708943950328983162243962653943
419778613112216264068985797800513224
0328602782204
$p =$ 1716199415032652428745475199770348 30
431735882503582635234861586479638579
584941427235987582245790560990663485
854944653278054066017246449450128306
0082150695551
$h =$ 4

$m =$ 569
$f =$ $x^{569}+x^{441}+x^{313}+x^{121}+1$
$d =$ $t^{56}+t^{45}+t^{41}+1$
$G_x =$ 195b22b2864ee08dd456bab1a95cdd8c7e3
fd330fddf630f9c3bb5c33f062b341c919c
6bb4cbf1d4335a344ed023b319585ea0e16
f03453cc5ba9a86a4b28b16e1c72ad75f11
41f
$G_y =$ 195b22b2864ee08df456bab1a95cdd8c5e3
fd330fddf630f9c3bb5c33f062b341c919c
6bb4cbf1d4135a344ed023b319785ea0e16
f03453ce5ba9a86a4b28b16e1c72ad75f11
41f
$G_{1/w} =$ 10000000000000001
$n =$ 19322687615086291723476759454659936 7
21494636648532174993286176257257595 7
11447802122680324312135204225690994 1
05833596555418474932773179050848492 2
46503400252208800671993093 08
$p =$ 48306719037715729308691898636649841 8
03736591621330437483215440643143989 2
78619505306700810780338010564227485 2
64583991388546187331932947627121230 6
16258500630522001679982732 7
$h =$ 4

For each curve, we selected the one with the best friendly generator among a set of generated curves. Computations were done on a cluster of 80 cores running Scientific Linux 6 during one and a half month generating 48 new curves of different sizes.