Generation of (Twisted Edwards) elliptic curves for circuit use

Barry WhiteHat[1], Jordi Baylina[2], and Marta Bellés[2,3]

[1]*Ethereum foundation,* [2]*iden3,* [3]*Universitat Pompeu Fabra*

April 18, 2019

Contents

1 Scope, motivation and background

Motivation:

- Protocols like zk-SNARK require the usage of pairing-friendly elliptic curves (references to SNARK, pairing-friendly elliptic curves and BN, BL).
- What happens if inside the circuit we need to implement protocols that use elliptic curves, such as Pedersen Hash and EdDSA.
- Inside the SNARK, we have elements of a certain field $\mathbb{F}_p$ (which is the order of the pairing-friendly curve, is at always of prime order?)
- So, we have to find what we call an *embedded curve* defined over $\mathbb{F}_p$.

Aim:

- This proposal aims to define a deterministic algorithm for generating elliptic curves defined over a given prime field. That is, given a prime p, generate an elliptic curve defined over $\mathbb{F}_p$.
- It is important the ability to find it in a deterministic way—so that it was clear no other considerations were taken for defining—is paramount as it significantly reduces the possibility of a backdoor being present, thus leading to better security.
- Also: provide an algorithm for checking its safety against best known attacks.
- We present an example of a generated curve done this way: Baby Jubjub. Also include the arithmetic implementation of it.

Background:

- Elliptic curve? Pairing-friendly (BN, BL)? Twisted Edwards / Montgomery elliptic curve? (Add REF).
- Order of the elliptic curve? Large prime dividing the order?
- Safety criteria: add attacks? (Add REF?)

Terminology:

- *Embedded elliptic curve*: given an elliptic curve E of prime order p, an elliptic curve E' defined over $\mathbb{F}_p$ is called an embedded elliptic curve of E.

Notation:

Notation	**Description**
E	Twisted Edwards elliptic curve.
p	Prime number. Characteristic of the field E is defined over.
a d	Parameters of the equation $ax^2 + y^2 = 1 + dx^2y^2$.
n	Order of the curve. Typically, $n = h \times l$.
l	Large prime dividing the order of the curve.
h	Cofactor.
x_0, y_0	Generator of the curve.
x_1, y_1	Generator of the large prime subgroup of the curve.

2 Generation of embedded elliptic curves

Definition of the curve:

- Finite field (defined by p).
- Order of the curve and its prime decomposition* (in case of TEd/Mont: cofactor and large prime).
- Generator and base point.

2.1 Twisted Edwards and Montgomery

In 2016, a group of researchers of IRPF designed a deterministic algorithm that, given a prime number p, it returns the elliptic curve defined over $\mathbb{F}_p$ with smallest coefficient A such that $A-2$ is a multiple of 4 and equation $y^2 = x^3 + Ax^2 + x$ describes a Montgomery curve. The assumption $A-2$ divisible by 4 comes from the fact that as this value is used in many operations, so trying to keep it smaller and divisible by four is a reasonable assumption [5].

2.1.1 Generation of elliptic curves given p with $p \equiv 1 \mod 4$

```
import sys
import pdb

def findCurve(prime, curveCofactor, twistCofactor, _A):
        F = GF(prime)
        A = _A
        while A < _A + 100000:
                print A
                if (A-2.) % 4 != 0:
                        A+=1.
                        continue
                try:
                        E = EllipticCurve(F, [0, A, 0, 1, 0])
                except:
                        A+=1.
                        continue
                groupOrder = E.order()
                if (groupOrder % curveCofactor != 0
                or not is_prime(groupOrder // curveCofactor)):
                        A+=1
                        continue

                twistOrder = 2*(prime+1)-groupOrder
                if (twistOrder % twistCofactor != 0
                or not is_prime(twistOrder // twistCofactor)):
                        A+=1
                        continue
                return A, E

def find1Mod4(prime, curveCofactor, twistCofactor, A):
        assert((prime % 4) == 1)
        return findCurve(prime, curveCofactor, twistCofactor, A)

def findGenPoint(prime, A, EC, N):
        F = GF(prime)
        for uInt in range(1, 1e3):
                u = F(uInt)
                v2 = u^3 + A*u^2 + u
                if not v2.is_square():
                        continue
                v = v2.sqrt()

                point = EC(u, v)
                pointOrder = point.order()
                if pointOrder == N:
                        return point

def mont_to_ted(u, v , r):
        x = Mod(u / v, r)
        y = Mod((u-1)/(u+1), r)
        return(x, y)

def ted_to_mont(x, y , r):
        u = Mod((1 + y )/ ( 1 - y)  , r)
        v = Mod((1 + y ) / ( (1 - y) * x) , r )
        return(u,v)

def isOnEd(x,y,r,a,d):
        return Mod(Mod(a,r)*(x**2),r) + Mod(y**2 , r) - 1
        - Mod(d,r)*(Mod(x**2,r))*(Mod(y**2,r)) == 0
```

2.1.2 Generation of elliptic curves given p with $p \equiv 3 \mod 4$

2.2 Elliptic curves of prime order

2.3 Pairing friendly elliptic curves

3 Safety criteria

SafeCurves is a project that checks some of the most common and known attacks on several elliptic curves. It also provides the algorithm it was used [3].

4 Example: embedded curve of BN128 generation of Baby Jubjub

4.1 Generation Of Baby Jubjub

We considered the large prime number dividing the order of BN128 and run algorithm A.1 from [5]. The first elliptic curve it was returned satisfying SafeCurves criteria was the Montgomery curve with coefficient $A = 168698$. We named this curve Baby Jubjub elliptic curve.

4.1.1 Code

```
prime = 21888242871839275222246405745257275088
        5483644004160343436982041865758084956 17
Fr = GF(prime)
h = 8 # cofactor
```

```
A = int(sys.argv[1])
A, EC = find1Mod4(prime, h, 4, A)

# A = 170214 another candidate
B = 1
a = A + 2 / B
d = A - 2 / B

print "a " , a , "d " , d , sqrt(d)
# check we have a safe twist
assert(not d.is_square())
assert(a*d*(a-d)!=0)

s = factor(EC.order())
print ("l : " , s)
N = h * s # order of the curve
print (factor(EC.quadratic_twist().order()))

# get generator point
u_gen, v_gen, w_gen = findGenPoint(prime, A, EC, N)
# find that generator point on the edwards curve
gen_x, gen_y = mont_to_ted(u_gen, v_gen, prime)
# make sure the generator point is on the twisted edwards curve
assert(isOnEd(gen_x, gen_y, prime, a , d))
# go back to montgomery
u , v = ted_to_mont(gen_x, gen_y, prime)
# confirm we are back where we started from
assert (u == u_gen)
assert (v == v_gen)

# get base point on montgorery curve by multiplying the generator point by h
base_x , base_y, base_z = h*EC(u_gen, v_gen)
# find the same points on twisted edwards curve
base_x , base_y = mont_to_ted(base_x , base_y, prime)

# the generator is on the twisted edwards curve
assert(isOnEd(base_x,base_y, prime , a , d))

print ("generator :" , gen_x, gen_y)
print ("base :", base_x, base_y)
```

4.2 Definition Of Baby Jubjub

From now on, let

$$p = 21888242871839275222246405745257275088548364400416034343698204186575808495617$$

and $\mathbb{F}_p$ the finite field with p elements.

4.2.1 Montgomery Form

We define E_M as the *Baby-Jubjub* Montgomery elliptic curve defined over $\mathbb{F}_p$ given by equation

$$E : v^2 = u^3 + 168698u^2 + u.$$

The order of E_M is $n = 8 \times l$, where

$$l = 2736030358979909402780800718157159386076813972158567259200215660948447373041$$

is a prime number. Denote by $\mathbb{G}$ the subgroup of points of order r, that is,

$$\mathbb{G} = \{ \, P \in E(\mathbb{F}_p) \mid lP = O \, \}.$$

4.2.2 Edwards Form

E_M is birationally equivalent to the Edwards elliptic curve

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

where $d = 9706598848417545097372247223557719406784115219466060233080913168975159366771$.

The birational equivalence [2, Thm. 3.2] from E to E_M is the map

$$(x, y) \to (u, v) = \left(\frac{1+y}{1-y}, \frac{1+y}{(1-y)x} \right)$$

with inverse from E_M to E

$$(u, v) \to (x, y) = \left(\frac{u}{v}, \frac{u-1}{u+1} \right).$$

4.2.3 Parameters

Notation	**Value**
p	21888242871839275222246405745257275088548364400416034343698204186575808495617
a	-1
d	12181644023421730124874158521699555681764249180949974110617291017600649128846
n	21888242871839275222246405745257275088614511777268538073601725287587578984328
l	2736030358979909402780800718157159386076813972158567259200215660948447373041
h	8
x_0	995203441582195749578291179787384436505546430278305826713579947235728471134
y_0	5472060717959818805561601436314318772137091100104008585924551046643952123905
x_1	5299619240641551281634865583518297030282874472190772894086521144482721001553
y_1	16950150798460657717958625567821834550301663161624707787222815936182638968203

4.3 Arithmetic In Baby Jubjub

In this section we define how to operate in the elliptic curve group: the addition of points and multiplication of a point by a scalar (an element of $\mathbb{F}_p$).

4.3.1 Addition Of Points

When adding points of elliptic curves in Montgomery form, one has to be careful if the points being added are equal (doubling) or not (adding) and if one of the points is the point at infinity [6]. Edwards curves have the advantage that there is no such case distinction and doubling can be performed with exactly the same formula as addition [2]. In comparison, operating in Montgomery curves is cheaper. In this section, we summarize how addition and doubling is performed in both forms. For the exact number of operations required in different forms of elliptic curves, see [2].

- Edwards: Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be points of the Baby-Jubjub twisted Edwards elliptic curve E. The sum $P_1 + P_2$ is a third point $P_3 = (x_3, y_3)$ with

$$\begin{aligned}
\lambda &= d x_1 x_2 y_1 y_2,\\
x_3 &= (x_1 y_2 + y_1 x_2)/(1+\lambda),\\
y_3 &= (y_1 y_2 - x_1 x_2)/(1-\lambda).
\end{aligned}$$

  Note that the neutral element is the point $O = (0, 1)$ and the inverse of a point (x, y) is $(-x, y)$.

- Montgomery: Let $P_1 = (x_1, y_1) \neq O$ and $P_2 = (x_2, y_2) \neq O$ be two points of the Baby-JubJub elliptic curve E_M in Montgomery form.

  If $P_1 \neq P_2$, then the sum $P_1 + P_2$ is a third point $P_3 = (x_3, y_3)$ with coordinates

$$\begin{aligned}
\Lambda &= (y_2 - y_1)/(x_2 - x_1),\\
x_3 &= \Lambda^2 - A - x_1 - x_2,\\
y_3 &= \Lambda(x_1 - x_3) - y_1.
\end{aligned} \tag{1}$$

  If $P_1 = P_2$, then $2 \cdot P_1$ is a point $P_3 = (x_3, y_3)$ with coordinates

$$\begin{aligned}
\Lambda &= (3x_1^2 + 2Ax_1 + 1)/(2y_1),\\
x_3 &= \Lambda^2 - A - 2x_1,\\
y_3 &= \Lambda(x_1 - x_3) - y_1.
\end{aligned} \tag{2}$$

4.3.2 Multiplication Of A Point Of E By A Scalar

Let $P \neq O$ be a point of the Edwards curve E of order strictly greater than 8 (i.e. $P \in \mathbb{G}$) and let k a binary number representing an element of $\mathbb{F}_p$. We describe the circuit used to compute the point $k \cdot P$.

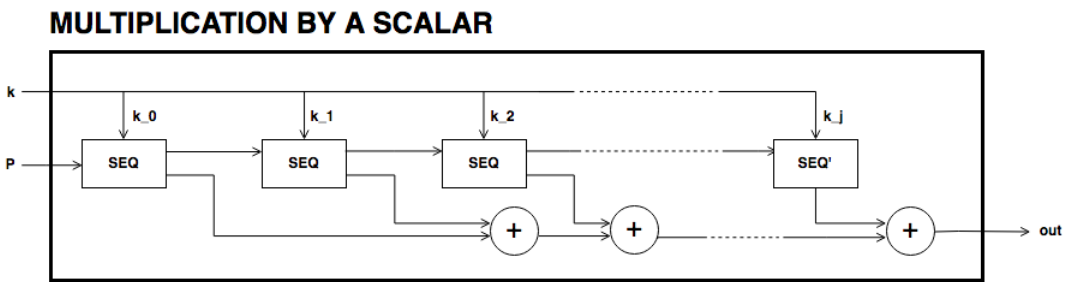
1. First, we divide k into chunks of 248 bits. If k is not a multiple of 248, we take j segments of 248 bits and leave a last chunk with the remaining bits. More precisely, write

$$k = k_0 k_1 \dots k_j \quad \text{with} \quad \begin{cases} k_i = b_0^i b_1^i \dots b_{247}^i & \text{for } i = 0, \dots, j-1,\\ k_j = b_0^j b_1^j \dots b_s^j & \text{with } s \leq 247. \end{cases}$$

   Then,

$$k \cdot P = k_0 \cdot P + k_1 \cdot 2^{248} P + \dots + k_j \cdot 2^{248j} P. \tag{3}$$

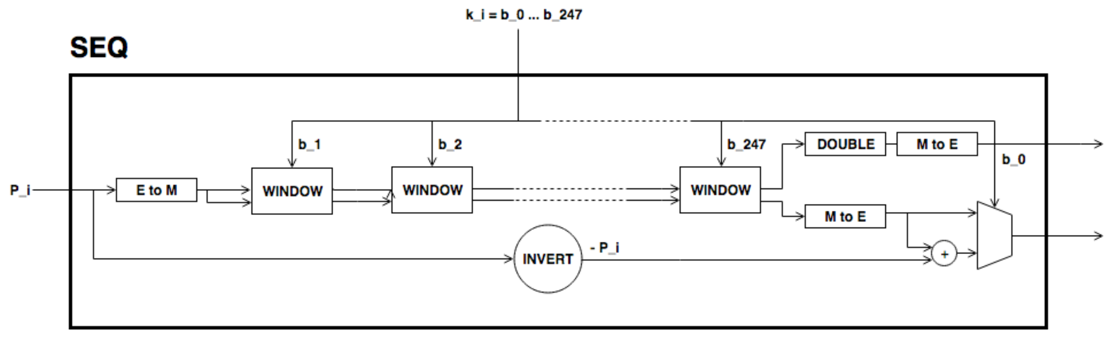
   This sum is done using the following circuit. The terms of the sum are calculated separately inside the SEQ boxes and then added together.

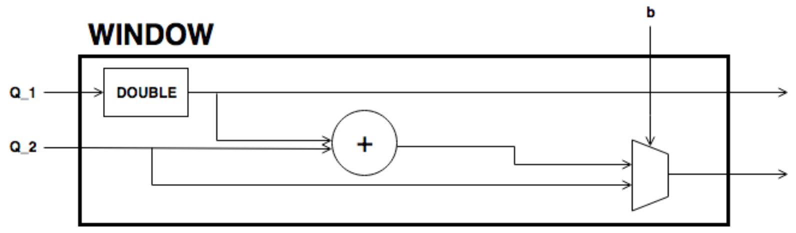


2. Each SEQ box takes a point of E of the from $P_i = 2^{248i}P$ for $i = 0, \ldots, j-1$ and outputs two points

$$2^{248} \cdot P_i \quad \text{and} \quad \sum_{n=0}^{247} b_n \cdot 2^n \cdot P_i.$$

The first point is the input of the next $(i+1)$-th SEQ box (note that $2^{248} \cdot P_i = P_{i+1}$) whereas the second output is the computation of the i-th term in expression (3). The precise circuit is depicted in next two figures SEQ and WINDOW.





The idea of the circuit is to first compute

$$Q = P_i + b_1 \cdot (2P_i) + b_2 \cdot (4P_i) + b_3 \cdot (8P_i) + \cdots + b_{247} \cdot (2^{247}P_i),$$

and output the point

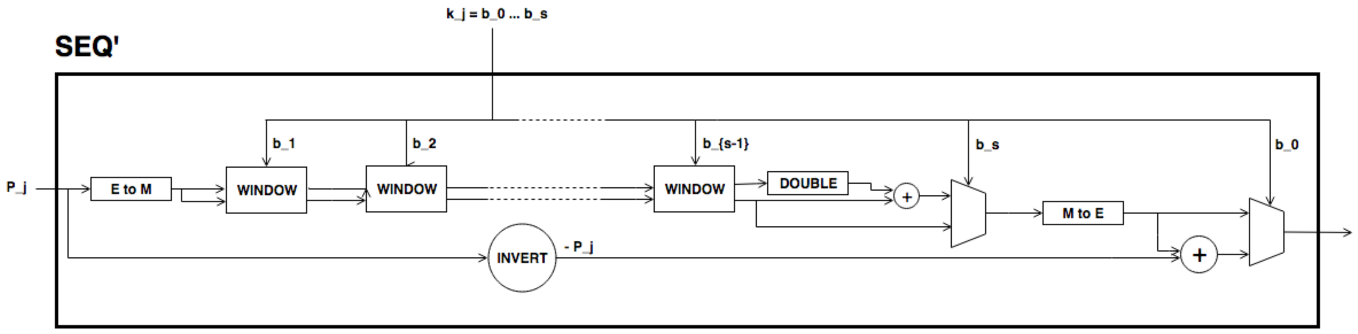
$$Q - b_0 \cdot P_i.$$

This permits the computation of Q using the Montgomery form of Baby-Jubjub and only use twisted Edwards for the second calculation. The reason to change forms is that, in the calculation of the output, we may get a sum with input the point at infinity if $b_0 = 0$.

Still, we have to ensure that none of the points being doubled or added when working in E_M is the point at infinity and that we never add the same two points.

- By assumption, $P \neq O$ and $\mathrm{ord}(P) > 8$. Hence, by Lagrange theorem [1, Corollary 4.12], P must have order r, $2r$, $4r$ or $8r$. For this reason, none of the points in E_M being doubled or added in the circuit is the point at infinity, because for any integer m, 2^m is never a multiple of r, even when 2^m is larger than r, as r is a prime number. Hence, $2^m \cdot P \neq O$ for any $m \in \mathbb{Z}$.
- Looking closely at the two inputs of the sum, it is easy to realize that they have different parity, one is an even multiple of P_i and the other an odd multiple of P_i, so they must be different points. Hence, the sum in E_M is done correctly.

3. The last term of expression (3) is computed in a very similar manner. The difference is that the number of bits composing k_j may be shorter and that there is no need to compute P_{j+1}, as there is no other SEQ box after this one. So, there is only output, the point $k_j \cdot P_j = k_j \cdot 2^{248j} P$. This circuit is named SEQ'.



5 Challenges And Security

As required in the construction of Baby-Jubjub, the curve satisfies SafeCurves criteria. This can be checked following [4].

6 Implementation

Barry WhiteHat:

- https://github.com/barryWhiteHat/baby_jubjub
- https://github.com/barryWhiteHat/baby_jubjub_ecc

Jordi Baylina: https://github.com/iden3/circomlib/blob/master/src/babyjub.js

7 Intellectual Property

We will release the final version of this proposal under creative commons, to ensure it is freely available to everyone.

References

[1] Baumslag, B., and Chandler, B. *Schaum's outline of Theory and Problems of Group Theory.* Schaum's outline series. McGraw-Hill Book Company, New York, 1968. http://poincare.matf.bg.ac.rs/~zarkom/Book_Shaums_Group_theory.pdf.

[2] Bernstein, D. J., Birkner, P., Joye, M., Lange, T., and Peters, C. Twisted edwards curves. Cryptology ePrint Archive, Report 2008/013, March 13, 2008. `https://eprint.iacr.org/2008/013`.

[3] Bernstein, D. J., and Lange, T. Safecurves: choosing safe curves for elliptic-curve cryptography. `https://safecurves.cr.yp.to`, Accessed February 25, 2018.

[4] Hat, B. W. Baby-jubjub supporting evidence. GitHub, October 19, 2018. `https://github.com/barryWhiteHat/baby_jubjub`.

[5] Langley, A., Hamburg, M., and Turner, S. Elliptic Curves for Security. RFC 7748, January, 2016. `https://rfc-editor.org/rfc/rfc7748.txt`.

[6] Okeya, K., Kurumatani, H., and Sakurai, K. Elliptic curves with the montgomery-form and their cryptographic applications. In *Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography* (London, UK, UK, 2000), PKC '00, Springer-Verlag, pp. 238–257.