An Encoding for the Theta Model of Elliptic Curves

Nafissatou Diarra[1(✉)] and Emmanuel Fouotsa[2]

[1] Cheikh Anta Diop University, Dakar, Senegal
fifiramatou@gmail.com
[2] The University of Bamenda, Bamenda, Cameroon
emmanuelfouotsa@yahoo.fr

Abstract. The use of elliptic curves in cryptography requires to be able to transform an information (generally a bit string) to a point of an elliptic curve. This transformation, called encoding, must be such that the encoded message can be easily and uniquely recovered from the corresponding point. In this paper we propose a new encoding that maps an element of $\mathbb{F}_q$ to a point on the theta model for elliptic curves $E_\lambda:\ 1+x^2+y^2+x^2y^2=\lambda^2xy$ recently introduced in [9]. In particular, we show that this new encoding is efficiently computable (deterministic and polynomial-time). We also present a Sage software implementation to ensure the correctness of the encoding on this curve.

Keywords: Theta model · Elliptic curves · Deterministic encoding

1 Introduction

Many elliptic curve-based cryptographic schemes require to hash into the group of points of an elliptic curve, such as password-based authentication protocols (SPEKE (Simple Password Exponential Key Exchange), PAK (Password Authenticated Key exchange)), as well as various signature schemes based on the hardness of the DLP (Discrete Logarithm Problem). The main idea for constructing a hash function into elliptic curves is the following: the image of a message (an arbitrary string) m by the hash function F is $F(m) = f(h(m))$, where h is a classical hash function and f is an encoding function that maps a point of $\mathbb{F}_q$ to an element of the curve. But a problem arises: given any elliptic curve E over any finite field $\mathbb{F}_q$, how to construct, in a *deterministic* way, a non-zero point of the curve? Some authors have proposed algorithms to answer this question. But before 2006, only probabilistic solutions were known. The paper [3] of Boney and Franklin in 2001 was one of the first that required hashing into (supersingular) elliptic curves; in fact, the public key of their identity-based

This work is supported by the Pole of research in Mathematics with Applications to Information Security (PRMAIS, SubSaharan Africa) sponsored by Simons Foundation and LIRIMA-MACISA Project.

C. M. F. Kebe et al. (Eds.): InterSol 2018, LNICST 249, pp. 224–235, 2018.
https://doi.org/10.1007/978-3-319-98878-8_21

encryption is a point on the curve. In 2006, Shallue and Van de Woestjine proposed the first algorithm [15] that maps in a deterministic way an element of $\mathbb{F}_q$ (with odd characteristic) to a point of any elliptic curve over $\mathbb{F}_q$. Their algorithm is based on the Skalba's equality theorem and requires to compute a square root in $\mathbb{F}_q$; but it can constructs only $(q-4)/8$ points of the curve.

In 2009, Icart defined a new encoding function [13] for Weierstrass form of elliptic curves, based on a very simple idea: intersect the line $y = ux + v$ with the equation of the curve. He showed that his algorithm works in $O \log^3(q)$ operations in $\mathbb{F}_q$ and conjectured (it was proven later by Tibouchi and Fouque [12]) that the size of the image set is approximately $\frac{5}{8}$ of the size of the curve.

Some authors have also proposed constructions of encoding functions for special families of elliptic curves, such as Hessian curves (by Farashahi in [10]), Edwards curves (elligator functions by Bernstein et al. in [1]), or Huff curves (by Diarra *et al.* in [8]).

Our goal on this paper is to continue this line of research, by proposing an encoding function for the theta model for elliptic curves $E_\lambda :\ 1+x^2+y^2+x^2y^2 = \lambda^2 xy$, recently introduced by Fouotsa and Diao [6]. In particular, we will show that this new encoding is efficiently computable (deterministic and polynomial-time).

The rest of the paper is structured as follows: In Sect. 2, we recall a special mathematical concept needed in the work. We briefly define elliptic curves and present the theta model for elliptic curves. together with an overview of main existing encodings into elliptic curves. The Sect. 3 describes our new encoding on the theta model and describes its properties. A numerical example is given with a code written with the Sage software to ensure the correctness of the encoding. We conclude our work in Sect. 4.

2 Preliminaries

2.1 Quadratic Character

Let $p \neq 2$ be a prime and $\mathbb{F}_{p^n} = \mathbb{F}_q$ the finite field of $q = p^n$ elements (where $n \geq 1$ is an integer). An element $a \in \mathbb{F}_q$ is a quadratic residue if there exists $r \in \mathbb{F}_q$ s.t. $a \equiv r^2 \mod q$. We define the quadratic character as follows: $\chi : \mathbb{F}_q \to \mathbb{F}_q : a \mapsto \chi(a) = a^{(q-1)/2}$; it verifies: $\chi(a) = 1$ if a is a non-zero quadratic residue, $\chi(a) = 0$ if $a = 0$ and $\chi(a) = -1$ otherwise. The following properties are also verified: $\chi(ab) = \chi(a) \cdot \chi(b)$ for any $a, b \in \mathbb{F}_q$; $\chi(a^2) = 1$ for any $a \in \mathbb{F}_q^*$; and if $q \equiv 3 \mod 4$, $\chi(-1) = -1,\ \chi(\chi(a)) = \chi(a)$, for any $a \in \mathbb{F}_q$. If $q \equiv 1 \mod 4$, then $\chi(-1) = 1$.

2.2 The Theta Model for Elliptic Curves

Elliptic Curves. An elliptic curve E over a field $\mathbb{K}$ is the set of solution in $\mathbb{A}^2(\overline{\mathbb{K}})$ of the equation

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \text{with } (a_1, a_2, a_3, a_4, a_6) \in \mathbb{K}^5 \quad (1)$$

together with a rational point $\mathcal{O}$ and the condition $\Delta \neq 0$ where $\Delta = -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6$ with $d_2 = a_1^2 + 4a_2, d_4 = 2a_4 + a_1, d_6 = a_3^2 + 4a_6, d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2$.

The quantity Δ is called the discriminant of E and the condition $\Delta_E \neq 0$ ensures that the curve E is smooth. In the set of point of an elliptic curve, it is very easy to set an additive group structure using the chord-and-tangent method (see [16] for complete details). When an elliptic curve is defined over a finite field, the resulting group presents a difficult mathematical problem known as the Discrete Logarithm Problem stated as follows: Given a point Q multiple of another given point P, find the integer n such that $Q = nP$. This problem justifies the use of elliptic curves in cryptography for the construction of several secure cryptosystems. The model of elliptic curve given by Eq. (1) is called the Weierstrass model and is the commonly used in the literature. Several other models exists in the literature such as the Edwards model [7], Hessian curves [14], Huff curves [5] and the Jacobi curves [2,4] including the theta model recently introduced by Fouotsa and Diao [6]. Although these curves are birationally equivalent to each other, in an algorithmic point of view and for security and efficiency purposes, careful choices need to be made on which model of elliptic to use in cryptography.

The Theta Model. This model of elliptic curves was proposed by Fouotsa and Diao [9]. The model is obtained from theta functions and the equation is given as follows $E_\lambda :\ 1 + x^2 + y^2 + x^2y^2 = \lambda^2xy$. They showed that their model is birationally equivalent to the Weierstrass model $v^2 = u^3 - (1 + c^4)u^2 - 4c^4u + 4c^4(1 + c^4)$. This model enjoys many other properties such as unified formulas (addition and doubling of points use the same formulas) and presents competitive addition formulas over binary fields. More details can be found in [6,9].

2.3 Existing Encodings for Elliptic Curves

In this section, we give a short overview of existing methods to encode into elliptic curves.

Trivial Encoding: For an elliptic curve $E_{a,b} :\ y^2 = x^3 + ax + b$ over the field $\mathbb{F}_q$, the simplest way to construct a point of $E_{a,b}$ from an element of $\mathbb{F}_q$ is to use the trivial encoding, also known as the *try-and-increment* method. The idea is to pick a x-coordinate and try to deduce the y-coordinate by computing a square root: choose a random element $u \in \mathbb{F}_q^*$ and compute $u^3 + au + b$; and then test whether $u^3 + au + b$ is a square in $\mathbb{F}_q$. If it is the case, then returns

$(x, y) = (u, \pm\sqrt{u^3 + au + b})$ as a point of the curve. Otherwise, one can choose another u in $\mathbb{F}_q$ and try again. But this method has at least one drawback as it cannot *run in constant time*: the number of operations depends on the input u. In practice the input u is the message m we want to hash; thus running this algorithm can allow the attacker to guess some information about m.

Icart's Encoding: Let $q \equiv 2 \mod 3$. The map $x \mapsto x^3$ is a bijection and then computation of a cubic root can be done as an exponentiation. In [13], Icart defined a new encoding function, based on the following idea: intersect the line $y = ux + v$ with the Weierstrass curve $E_{a,b} : \ y^2 = x^3 + ax + b$, with $a, b \in \mathbb{F}_q$. He defined the encoding function:

$$\begin{aligned} f_{a,b} : \mathbb{F}_q &\to E_{a,b} \\ u &\mapsto f_{a,b}(u) = (x, ux + v) \end{aligned}$$

where $x = (v^2 - b - \frac{u^6}{27})^{1/3} + \frac{u^2}{3}$ and $v = (3a - u^4)/6u$.

As shown in the paper, this function presents many interesting properties. In fact, it can be implemented in polynomial time with $O(\log^3 q)$ operations. The inverse function $f_{a,b}^{-1}$ is also computable in polynomial time. Icart also showed that $|f_{a,b}^{-1}(P)| \leq 4$, given a point P on the elliptic curve.

Other Existing Encodings: There exist many other encodings for special families of elliptic curves, such as supersingular curves (by Boneh and Franklin in [3]), Hessian curves (by Farashahi in [10]), Edwards curves (Elligator functions by Bernstein *et al.* in [1]), Huff curves (by Diarra *et al.* in [8]), etc.

3 A New Encoding for the Theta Model

3.1 The Algorithm

In this section, we propose a deterministic algorithm that given an element r (with additional conditions) of $\mathbb{F}_q$, constructs a point on $E_\lambda(\mathbb{F}_q)$. From this algorithm, we define the new encoding function which does not cover all points of $\mathbb{F}_q$, unless we make some additional hypothesis on the underlying field $\mathbb{F}_q$. Nevertheless, we can send all elements of $\mathbb{F}_q$ that are not in the set of definition (there are at most 6 such points) of the encoding to the point at infinity.

Algorithm 1. Encode-Theta-Model

Input : q be a prime power, $c \in \mathbb{F}_q$ s.t. $c(1-c^4)(1+c^4) \neq 0$, $u \in \mathbb{F}_q$ s.t. $\chi(u) = -1$, $r \in \mathcal{R} = \{r \in \mathbb{F}_q : \; 4ur^2c^6 \neq (1-c^4)(1+3c^4),\; 4ur^2c^6 \neq -(1-c^4)^4, (4ur^2c^6)(1+3c^4) \neq (1-c^4)^3 \;\} \subseteq \mathbb{F}_q$

Output: A point (x_λ, y_λ)

$$v = c \cdot \left(\frac{4ur^2c^6 + (1-c^4)(3+c^4)}{4ur^2c^6 + (1-c^4)(-3c^4-1)}\right);$$

$$\varepsilon = \chi\left((c^2-v^2)(1-c^2v^2)\right);$$

$$X = \frac{1}{2}\left((1+\varepsilon)v + (1-\varepsilon)\left(\frac{(-c^4-1)(v+c) - c(1-c^4)}{2c^3(v+c)+1-c^4}\right)\right);$$

$$Y = -\varepsilon\sqrt{\frac{c^2-X^2}{1-c^2X^2}};$$

$$x_\lambda = \frac{X+1}{X-1};$$

$$y_\lambda = \frac{Y-1}{Y+1};$$

return (x_λ, y_λ);

Theorem 1. *The output (x_λ, y_λ) of Algorithm 1 is a point of the curve E_λ : $1+x^2+y^2+x^2y^2 = \lambda^2 xy$, where $\lambda^2 = \dfrac{4(1+c^2)}{1-c^2}$ and $\lambda(\lambda^2-4)(\lambda^2+1) \neq 0$.*

Proof. 1. v is well-defined from the definition of $\mathcal{R}$.

2. Let us show that $\varepsilon \neq 0$. Suppose that $\varepsilon = 0 \Leftrightarrow \mathbf{c^2 = v^2}$ or $\mathbf{c^2v^2 = 1}$.
   (a) $\mathbf{c^2 = v^2} \Rightarrow c = \pm v \Rightarrow 4(1-c^4)(1+c^4) = 0$ (impossible by the choice of c) or $4ur^2c^6 = (1-c^4)^2$ (impossible since u is not a square).
   (b) $\mathbf{c^2v^2 = 1} \Rightarrow cv = \pm 1 \Rightarrow 4ur^2c^6 = (c^2+1)^4$ or $4ur^2c^6 = (c-1)^4(c+1)^4$; this is impossible since u is not a square.

   So $\varepsilon \neq 0$ and then $\varepsilon = \pm 1$.
3. Let us show that X, Y are well-defined. For this, we consider the two cases $\varepsilon = 1$, $\varepsilon = -1$ and show that the quantity $\dfrac{c^2-X^2}{1-c^2X^2}$ is a square.
   $\varepsilon = 1 \Rightarrow X = v$ and $\chi\left(\dfrac{c^2-X^2}{1-c^2X^2}\right) = \chi\left(\dfrac{c^2-v^2}{1-c^2v^2}\right) = \varepsilon = 1$; so X and Y are well-defined.
   $\varepsilon = -1 \Rightarrow X = -\dfrac{(-c^4-1)(v+c) - c(1-c^4)}{2c^3(v+c)+1-c^4}$. Now let $H(X) = \dfrac{c^2-X^2}{1-c^2X^2} = \dfrac{(c-X)(c+X)}{1-c^2X^2}$; we want to express $H(X)$ in term of $H(v)$ and use the value $\chi(H(v)) = \chi\left(\dfrac{c^2-v^2}{1-c^2v^2}\right) = \varepsilon = -1$. Now to have an expression of $H(X)$ in terms of $H(v)$, we compute separately the value of $c-X, c+X$ and $1-c^2X^2$ and find that:

$\left[2c^3(v+c)+(1-c^4)\right](c-X)=(v+c)(c^4-1)$,
$\left[2c^3(v+c)+(1-c^4)\right](c+X)=(v+c)(3c^4+1)+2c(1-c^4)$,
and $\left[2c^3(v+c)+(1-c^4)\right]^2(1-c^2X^2)=(1-c^2v^2)(c^4-1)^2$. This leads to

$$\begin{aligned} H(X) = &\quad \frac{(v+c)(c^4-1)\left[(v+c)(3c^4+1)+2c(1-c^4)\right]}{(1-c^2v^2)(c^4-1)^2} \\ = &\quad \frac{H(v)}{c-v}\left(\frac{(v+c)(3c^4+1)+2c(1-c^4)}{c^4-1}\right) \end{aligned}$$

Since $(v+c)(3c^4+1)+2c(1-c^4)=2c\left(\frac{8ur^2c^6(1+c^4)}{4ur^2c^6+(1-c^4)(-3c^4-1)}\right)$ and $(c-v)(c^4-1)=4c\left(\frac{(1-c^4)^2(1+c^4)}{4ur^2c^6+(1-c^4)(-3c^4-1)}\right)$, then we can rewrite $H(X)$ as follows:

$$H(X)=H(v)\left(\frac{4ur^2c^6}{(1-c^4)^2}\right)$$

and thus $\chi(H(X))=\chi(H(v))\cdot\chi\left[(4ur^2c^6)(1-c^4)^2\right]=-\chi(u)=1$. In other words, $\frac{c^2-X^2}{1-c^2X^2}$ is a square and Y is well-defined.

4. Since X and Y are well-defined (for both cases $\varepsilon=1$ and $\varepsilon=-1$), we can compute $Y^2=\frac{c^2-X^2}{1-c^2X^2}\Rightarrow X^2+Y^2=c^2(1+X^2Y^2)$. Now we have to show that x_λ and y_λ verify the relation $1+x_\lambda^2+y_\lambda^2+x_\lambda^2y_\lambda^2-\lambda^2x_\lambda y_\lambda=0$, where $\lambda^2=\frac{4(1+c^2)}{1-c^2}$. In fact, we have:
$1+x_\lambda^2+y_\lambda^2+x_\lambda^2y_\lambda^2-\lambda^2x_\lambda y_\lambda=1+\left(\frac{X+1}{X-1}\right)^2+\left(\frac{Y-1}{Y+1}\right)^2+\left(\frac{X+1}{X-1}\right)^2\left(\frac{Y-1}{Y+1}\right)^2-\lambda^2\left(\frac{X+1}{X-1}\right)\left(\frac{Y-1}{Y+1}\right)$
$=\frac{1}{(X-1)^2(Y+1)^2}\left[2(X^2+1)((Y+1)^2+(Y-1)^2)-\lambda^2(X^2-1)(Y^2-1)\right]$
$=\frac{1}{(X-1)^2(Y+1)^2}\left[4(1+X^2Y^2+c^2(1+X^2Y^2))-\frac{4(1+c^2)}{1-c^2}\left(1+X^2Y^2-c^2(1+X^2Y^2)\right)\right]$
$=\frac{4}{(1-c^2)(X-1)^2(Y+1)^2}\left[(1-c^2)(1+X^2Y^2+c^2(1+x^2Y^2))-(1+c^2)(1+X^2Y^2-c^2(1+x^2Y^2))\right]$
$=\frac{4}{(1-c^2)(X-1)^2(Y+1)^2}\left[2c^2(1+X^2Y^2)-2c^2(1+X^2Y^2)-c^4(1+X^2Y^2)+c^4(1+X^2Y^2)\right]$
$=0$. Moreover λ verifies $\lambda(\lambda^2-4)(\lambda^2+1)\neq 0$ from the conditions on c.

Definition 1. *The encoding function for the theta model for elliptic curves is the function*

$$\begin{aligned} f_\lambda\ :\ \mathcal{R}\subseteq\mathbb{F}_q &\to E_\lambda(\mathbb{F}_q) \\ r &\mapsto (x_\lambda,y_\lambda), \end{aligned}$$

where x_λ and y_λ are defined by Algorithm 1. If $r\in\mathbb{F}_q\setminus\mathcal{R}$, we set $f_\lambda(r)=\mathcal{O}_o$.

3.2 Size of the Set $\mathcal{R}$

- Our encoding covers a subset $\mathcal{R}$ of $\mathbb{F}_q$; this means that only elements of $\mathcal{R}$ can be encoded. And from the definition of $\mathcal{R}$, it is easy to see that at most 6 elements of $\mathbb{F}_q$ can not be encoded, that is $card(\mathbb{F}_q \setminus \mathcal{R}) \leq 6$. Since in practice q (the size of the field) is much greater than 6, thus we can state that our encoding f_λ covers a great proportion of $\mathbb{F}_q$. For example, for $q = 503$ (see the Appendix for the complete example), we find that $card(\mathcal{R}) = 501$ and thus f covers more than 99% of $\mathbb{F}_q$.
- To cover $\mathbb{F}_q$ (that is $\mathcal{R} = \mathbb{F}_q$), one can choose an element $c \in \mathbb{F}_q$ such that $\chi\left((1-c^4)(1+3c^4)\right) = 1$ and -1 is a square (for example when $q \equiv 2 \mod 3$), and then $\mathcal{R} = \mathbb{F}_q$.

Remark 1. – The choice of c (and u) does not have any impact on the running-time of the algorithm, since one must choose a suitable c (and u) before starting the algorithm. When $q \equiv 3 \mod 4$, one can choose $u = -1$; if $q \equiv 5 \mod 8$, one can choose $u = 2$.
- Compared to many existing encodings, we do not put any requirements on q (the authors of [13] proposed for example to choose $q \equiv 2 \mod 3$, in order to compute efficiently cubic roots).

3.3 Properties of Our Encoding

Lemma 1 (Polynomial time).
The function f_λ can be implemented in deterministic polynomial time, with approximately $O(\log^3(q))$ operations over $\mathbb{F}_q$.

Proof. – The function f_λ is deterministic in the sense that, once the parameters q, c and u are fixed, then any input $r \in \mathcal{R}$ will always give the same output $P = (x_\lambda, y_\lambda)$. In fact, the algorithm does not involve any random value.
- To show that the algorithm is also computable in polynomial time, we must evaluate its complexity. Globally, the computation of $P = (x_\lambda, y_\lambda)$ requires some inversions, some multiplications, one computation of the quadratic character χ and one square root computation. The computation of χ can be replaced by an exponentiation (to test if a is square, just compute a to the exponent $(q-1)/2$), which requires $O(\log^3(q))$ operations. Computing a square root in $\mathbb{F}_q$ requires $O(\log^3(q))$ operations when $q \equiv 3 \mod 4$; and more generally, it can be done in probabilistic polynomial time by using the Tonelli-Shanks algorithm. The inversions can be made efficiently by using extended Euclid algorithm or avoided by using projective coordinates (excepted for the last two inversions in x_λ and y_λ). So globally, we can expect our algorithm to run in polynomial time (with approximatively $O(\log^3(q))$ operations). □

From definition (1) and from Algorithm (1), it is easy to see that given any point $P \in \mathrm{Im}(f_\lambda)$ such that $f_\lambda(r) = P$ (for a certain $r \in \mathcal{R}$), we have $f_\lambda^{-1}(P) = \{r, -r\}$. This results from the fact that the definition of the encoding f_λ only involves r^2. Hence, if $f_\lambda(r) = P$, then $f_\lambda(-r) = P$ also. And one can show that $r, -r$ are the only points in the set $f_\lambda^{-1}(P)$ (like in [1] or in [8], this property of f_λ is called *almost-injectivity*). Moreover, we can invert f_λ as follows.

Lemma 2 (Inverting f_λ).
Given a point $P = (x_\lambda, y_\lambda) \in Im(f_\lambda)$, we can compute its preimage $r \in \mathcal{R}$ as follows:

– *if $\chi(y_\lambda) = -1$, then* $r = \dfrac{1}{2c^3}\sqrt{\dfrac{c(x-1)(3+c^4)+(1+x)(1+3c^4)}{u[x(1-c)+1+c]}}$;
– *if $\chi(y_\lambda) = 1$, then* $r = \dfrac{1}{2c^3(1-c^4)}\sqrt{\dfrac{(1+x)(5c^4-1)+c(1-x)(3+c^4)}{u[(1-c^4)(1+x+c(x-1))]}}$.

The proof is similar to those in [1,8].

Remark 2. Encodings into elliptic curves can be used in several ways. For example, Bernstein *et al.* [1] used an almost-injective encoding, namely Elligator-2, to make uniform strings indifferentiable from random. When the encoding function is well-distributed, it can be used to design an indifferentiable hash function into $E(\mathbb{F}_q)$. We can use these two applications for our encoding, since it is:

– *almost-injective:* injective when restricted to a certain subset S of $\mathbb{F}_q$. In fact, we can characterize the image set of f_λ and show that given a point P in $\mathrm{Im}(f_\lambda)$, $f_\lambda^{-1}(P) \in \{r, -r\}$ for some $r \in \mathbb{F}_q$. When $\mathbb{F}_q$ is a prime field, one can just set $S = \{0, 1, \ldots, \frac{q-1}{2}\}$;
– and *well-distributed:* in [11], Farashahi *et al.* showed that any deterministic encoding into elliptic curves can be transformed into a well-distributed one.

Example 1. We consider an example with the following parameters: $q = 503$, we set $u = -1$ and $c = 3$. A code for the implementation is given in appendix as well as the outputs for this example.

4 Conclusion

In this work, we described the first known encoding for the theta model for elliptic curves $E_\lambda : \; 1 + x^2 + y^2 + x^2y^2 = \lambda^2 xy$. And we showed that this new encoding is efficiently computable (deterministic and polynomial-time). A numerical example is also given to ensure the correctness of our encoding. Like existing encodings for other models of curves, our encoding has some interesting features, like almost-injectivity and inversibility. Such properties can be used to design indifferentiable hash functions in the group of points of the curve, or to design IBE-schemes.

A An Implementation of Theta-Model-Encoding in Sage

```
class ThetaModel():
    def init(self,q,u,c): ##assuming q is prime
        self.F=FiniteField(q,'a')
        self.q=q
        self.u=u
        self.c=c
    def verificationParameters(self):
        F=self.F
        q=self.q
        u=self.u
        c=self.c
        if F.characteristic()==2:
            print 'Error: The characteristic is egal to', F.characteristic()
            return False
        else:
            if (F(c).is_zero() is True) or (F(1-c^4).is_zero() is True) or (F(1+c^4).is_zero() is True):
                print 'Error: bad value for c'
                return False
            else:
                if (F(u).is_zero() is True) or (F(u).is_square() is True):
                    print 'Error: u={} is zero or is a square'.format(u)+' in the finite field of {} elements'.format(F.order())
                    return False
                else:
                    return True
    def setOfDefinition(self,value):
        F=self.F
        c=self.c
        u=self.u
        r=F(value)
        if (F(4*u*r*r*(c^6)-(1-c^4)*(1+3*c^4)).is_zero() is True) or
        ( F(4*u*r*r*(c^6)+(1-c^4)^4).is_zero() is True ) or ( F(4*u*r*r*(c^6)*(1+3*(c^4))-(1-c^4)^3).is_zero() is True):
            print 'r={} is not in the set of definition'.format(r)
            return False
        else:
            return True
    def quadraticCharacter(self,value):
        F=self.F
        try:
            residu=value.is_square()
        except:
            residu=F(value).is_square()
        if residu is True:
            return 1
        else:
            return -1
    def encodeTheta(self,value):
        if self.verificationParameters() is True:
            if self.setOfDefinition(value) is True:
                F=self.F
                r=F(value)
                c=self.c
                u=self.u
                v=F(  c*(4*u*r*r*(c^6)+(1-c^4)*(3+c^4))/(4*u*r*r*(c^6)+(1-c^4)*(-3*(c^4)-1))  );
                e=self.quadraticCharacter((c*c-v*v)*(1-c*c*v*v));
                X=F(  v*(e+1)/2  + (((-c^4-1)*(v+c)-c*(1-c^4))/(2*(c^3)*(v+c)+1-c^4))*(e-1)/2 );
                try:
                    a=F((c*c-X*X)/(1-c*c*X*X))
                    root=a.square_root()
                    Y=-e*root
                    x=(X+1)/(X-1)
                    y=(Y-1)/(Y+1)
                    return (F(x),F(y))
                except:
                    print 'Error when computing the square root of f(x)'
                    return {}
```

B Example with $q = 503, u = -1, c = 3$:.

```
t=ThetaModel()
t.init(501,-1,3)
if t.verificationParameters() is True:
    for i in t.F:
        if t.setOfDefinition(i) is True:
            print 'r=',i,'=====>(x,y)=',
            t.encodeTheta(i)
```

```
r= 0 =====>(x,y)= (430, 121)
r= 1 =====>(x,y)= (441, 382)
r= 2 =====>(x,y)= (386, 240)
r= 3 =====>(x,y)= (283, 274)
r= 4 =====>(x,y)= (212, 73)
r= 5 =====>(x,y)= (307, 100)
r= 6 =====>(x,y)= (23, 298)
r= 7 =====>(x,y)= (42, 171)
r= 8 =====>(x,y)= (311, 239)
r= 9 =====>(x,y)= (491, 332)
r= 10 =====>(x,y)= (415, 385)
r= 11 =====>(x,y)= (125, 490)
r= 12 =====>(x,y)= (330, 470)
r= 13 =====>(x,y)= (328, 205)
r= 14 =====>(x,y)= (315, 483)
r= 15 =====>(x,y)= (77, 31)
r= 16 =====>(x,y)= (352, 369)
r= 17 =====>(x,y)= (283, 123)
r= 18 =====>(x,y)= (327, 188)
r= 19 =====>(x,y)= (411, 365)
r= 20 =====>(x,y)= (480, 265)
r= 21 =====>(x,y)= (25, 485)
r= 22 =====>(x,y)= (386, 153)
r= 23 =====>(x,y)= (176, 404)
r= 24 =====>(x,y)= (368, 142)
r= 25 =====>(x,y)= (121, 62)
r= 26 =====>(x,y)= (40, 422)
r= 27 =====>(x,y)= (461, 50)
r= 28 =====>(x,y)= (318, 46)
r= 29 =====>(x,y)= (170, 76)
```

```
r= 30 =====>(x,y)= (217, 310)
r= 31 =====>(x,y)= (171, 12)
r= 32 =====>(x,y)= (187, 49)
r= 33 =====>(x,y)= (436, 313)
r= 34 =====>(x,y)= (31, 98)
r= 35 =====>(x,y)= (43, 350)
r= 36 =====>(x,y)= (382, 73)
r= 37 =====>(x,y)= (490, 125)
r= 38 =====>(x,y)= (338, 435)
r= 39 =====>(x,y)= (124, 141)
r= 40 =====>(x,y)= (378, 387)
r= 41 =====>(x,y)= (416, 457)
r= 42 =====>(x,y)= (63, 32)
r= 43 =====>(x,y)= (86, 51)
r=44 is not in the set of definition
r= 45 =====>(x,y)= (470, 330)
r= 46 =====>(x,y)= (259, 352)
r= 47 =====>(x,y)= (153, 460)
r= 48 =====>(x,y)= (318, 339)
r= 49 =====>(x,y)= (237, 69)
r= 50 =====>(x,y)= (220, 380)
r= 51 =====>(x,y)= (442, 330)
r= 52 =====>(x,y)= (213, 166)
r= 53 =====>(x,y)= (495, 110)
r= 54 =====>(x,y)= (393, 8)
r= 55 =====>(x,y)= (185, 164)
r= 56 =====>(x,y)= (287, 364)
r= 57 =====>(x,y)= (378, 13)
r= 58 =====>(x,y)= (472, 405)
r= 59 =====>(x,y)= (232, 304)
r= 60 =====>(x,y)= (51, 310)
r= 61 =====>(x,y)= (216, 139)
r= 62 =====>(x,y)= (258, 199)
r= 63 =====>(x,y)= (60, 476)
r= 64 =====>(x,y)= (258, 91)
r= 65 =====>(x,y)= (382, 441)
r= 66 =====>(x,y)= (352, 259)
r= 67 =====>(x,y)= (350, 117)
r= 68 =====>(x,y)= (109, 149)
r= 69 =====>(x,y)= (102, 336)
r= 70 =====>(x,y)= (467, 375)
r= 71 =====>(x,y)= (425, 154)
r= 72 =====>(x,y)= (339, 318)
r= 73 =====>(x,y)= (231, 209)
r= 74 =====>(x,y)= (379, 362)
r= 75 =====>(x,y)= (116, 334)
r= 76 =====>(x,y)= (252, 502)
r= 77 =====>(x,y)= (149, 109)
r= 78 =====>(x,y)= (235, 278)
r= 79 =====>(x,y)= (466, 338)
r= 80 =====>(x,y)= (457, 416)
r= 81 =====>(x,y)= (16, 123)
r= 82 =====>(x,y)= (495, 471)
r= 83 =====>(x,y)= (141, 215)
r= 84 =====>(x,y)= (470, 157)
r= 85 =====>(x,y)= (127, 94)
r= 86 =====>(x,y)= (334, 490)
r= 87 =====>(x,y)= (311, 181)
r= 88 =====>(x,y)= (301, 409)
r= 89 =====>(x,y)= (458, 67)
r= 90 =====>(x,y)= (440, 471)
r= 91 =====>(x,y)= (360, 253)
r= 92 =====>(x,y)= (350, 43)
r= 93 =====>(x,y)= (304, 245)
r= 94 =====>(x,y)= (322, 192)
r= 95 =====>(x,y)= (173, 33)
r= 96 =====>(x,y)= (18, 342)
r= 97 =====>(x,y)= (410, 179)
r= 98 =====>(x,y)= (485, 25)
r= 99 =====>(x,y)= (190, 488)
r= 100 =====>(x,y)= (475, 161)
r= 101 =====>(x,y)= (225, 122)
r= 102 =====>(x,y)= (8, 393)
r= 103 =====>(x,y)= (73, 382)
r= 104 =====>(x,y)= (342, 18)
r= 105 =====>(x,y)= (69, 237)
r= 106 =====>(x,y)= (442, 157)
r= 107 =====>(x,y)= (443, 27)
r= 108 =====>(x,y)= (403, 290)
r= 109 =====>(x,y)= (12, 453)
r= 110 =====>(x,y)= (148, 214)
r= 111 =====>(x,y)= (291, 430)
r= 112 =====>(x,y)= (289, 486)
r= 113 =====>(x,y)= (135, 361)
r= 114 =====>(x,y)= (250, 401)
r= 115 =====>(x,y)= (369, 493)
r= 116 =====>(x,y)= (305, 376)
r= 117 =====>(x,y)= (167, 143)
r= 118 =====>(x,y)= (493, 369)
r= 119 =====>(x,y)= (212, 441)
r= 120 =====>(x,y)= (125, 116)
r= 121 =====>(x,y)= (38, 235)
r= 122 =====>(x,y)= (490, 334)
r= 123 =====>(x,y)= (196, 337)
r= 124 =====>(x,y)= (263, 43)
r= 125 =====>(x,y)= (324, 93)
r= 126 =====>(x,y)= (357, 77)
r= 127 =====>(x,y)= (337, 290)
r= 128 =====>(x,y)= (337, 196)
r= 129 =====>(x,y)= (164, 185)
r= 130 =====>(x,y)= (460, 153)
r= 131 =====>(x,y)= (14, 224)
r= 132 =====>(x,y)= (245, 412)
r= 133 =====>(x,y)= (110, 495)
r= 134 =====>(x,y)= (271, 91)
r= 135 =====>(x,y)= (478, 18)
r= 136 =====>(x,y)= (151, 134)
r= 137 =====>(x,y)= (435, 189)
r= 138 =====>(x,y)= (467, 279)
r= 139 =====>(x,y)= (404, 483)
r= 140 =====>(x,y)= (489, 279)
r= 141 =====>(x,y)= (409, 301)
r= 142 =====>(x,y)= (493, 259)
r= 143 =====>(x,y)= (315, 176)
r= 144 =====>(x,y)= (28, 478)
r= 145 =====>(x,y)= (237, 226)
r= 146 =====>(x,y)= (157, 442)
r= 147 =====>(x,y)= (465, 268)
r= 148 =====>(x,y)= (394, 354)
r= 149 =====>(x,y)= (460, 240)
r= 150 =====>(x,y)= (198, 202)
r= 151 =====>(x,y)= (202, 198)
r= 152 =====>(x,y)= (189, 466)
r= 153 =====>(x,y)= (87, 339)
r= 154 =====>(x,y)= (78, 454)
r= 155 =====>(x,y)= (240, 386)
r= 156 =====>(x,y)= (213, 100)
r= 157 =====>(x,y)= (143, 250)
r= 158 =====>(x,y)= (224, 36)
r= 159 =====>(x,y)= (51, 86)
r= 160 =====>(x,y)= (435, 338)
r= 161 =====>(x,y)= (215, 132)
r= 162 =====>(x,y)= (411, 390)
r= 163 =====>(x,y)= (376, 305)
r= 164 =====>(x,y)= (275, 355)
r= 165 =====>(x,y)= (316, 349)
r= 166 =====>(x,y)= (307, 166)
r= 167 =====>(x,y)= (272, 294)
r= 168 =====>(x,y)= (502, 252)
r= 169 =====>(x,y)= (485, 161)
r= 170 =====>(x,y)= (463, 118)
r= 171 =====>(x,y)= (381, 38)
r= 172 =====>(x,y)= (217, 86)
r= 173 =====>(x,y)= (476, 60)
r= 174 =====>(x,y)= (264, 317)
r= 175 =====>(x,y)= (333, 427)
r= 176 =====>(x,y)= (88, 81)
r= 177 =====>(x,y)= (298, 175)
r= 178 =====>(x,y)= (67, 458)
r= 179 =====>(x,y)= (281, 119)
r= 180 =====>(x,y)= (338, 466)
r= 181 =====>(x,y)= (199, 258)
r= 182 =====>(x,y)= (362, 288)
r= 183 =====>(x,y)= (128, 14)
r= 184 =====>(x,y)= (384, 222)
r= 185 =====>(x,y)= (98, 357)
r= 186 =====>(x,y)= (290, 403)
r= 187 =====>(x,y)= (466, 189)
r= 188 =====>(x,y)= (186, 239)
r= 189 =====>(x,y)= (104, 69)
r= 190 =====>(x,y)= (36, 128)
r= 191 =====>(x,y)= (132, 124)
r= 192 =====>(x,y)= (46, 87)
r= 193 =====>(x,y)= (225, 268)
r= 194 =====>(x,y)= (263, 117)
r= 195 =====>(x,y)= (475, 25)
r= 196 =====>(x,y)= (487, 229)
r= 197 =====>(x,y)= (404, 176)
r= 198 =====>(x,y)= (328, 265)
r= 199 =====>(x,y)= (82, 390)
r= 200 =====>(x,y)= (104, 226)
r= 201 =====>(x,y)= (229, 220)
r= 202 =====>(x,y)= (488, 190)
r= 203 =====>(x,y)= (342, 28)
r= 204 =====>(x,y)= (161, 475)
r= 205 =====>(x,y)= (334, 116)
r= 206 =====>(x,y)= (489, 375)
r= 207 =====>(x,y)= (480, 205)
r= 208 =====>(x,y)= (403, 196)
r= 209 =====>(x,y)= (380, 487)
r= 210 =====>(x,y)= (278, 381)
r= 211 =====>(x,y)= (175, 238)
r= 212 =====>(x,y)= (375, 467)
r= 213 =====>(x,y)= (401, 167)
r= 214 =====>(x,y)= (288, 371)
r= 215 =====>(x,y)= (87, 46)
r= 216 =====>(x,y)= (271, 199)
r= 217 =====>(x,y)= (371, 379)
r= 218 =====>(x,y)= (50, 491)
r= 219 =====>(x,y)= (186, 181)
r= 220 =====>(x,y)= (10, 244)
r= 221 =====>(x,y)= (146, 426)
r= 222 =====>(x,y)= (31, 77)
r= 223 =====>(x,y)= (117, 263)
r= 224 =====>(x,y)= (440, 110)
r= 225 =====>(x,y)= (189, 435)
r= 226 =====>(x,y)= (502, 2)
r= 227 =====>(x,y)= (441, 212)
r= 228 =====>(x,y)= (465, 122)
r= 229 =====>(x,y)= (279, 489)
r= 230 =====>(x,y)= (483, 315)
r= 231 =====>(x,y)= (17, 228)
r= 232 =====>(x,y)= (91, 271)
r= 233 =====>(x,y)= (94, 127)
r= 234 =====>(x,y)= (169, 387)
r= 235 =====>(x,y)= (453, 42)
r= 236 =====>(x,y)= (15, 45)
r= 237 =====>(x,y)= (116, 125)
r= 238 =====>(x,y)= (20, 99)
r= 239 =====>(x,y)= (357, 98)
r= 240 =====>(x,y)= (310, 217)
r= 241 =====>(x,y)= (346, 61)
r= 242 =====>(x,y)= (226, 104)
r= 243 =====>(x,y)= (73, 212)
r= 244 =====>(x,y)= (471, 440)
r= 245 =====>(x,y)= (169, 13)
r= 246 =====>(x,y)= (412, 232)
r= 247 =====>(x,y)= (32, 63)
r= 248 =====>(x,y)= (332, 461)
r= 249 =====>(x,y)= (478, 28)
r= 250 =====>(x,y)= (82, 365)
r= 251 =====>(x,y)= (16, 274)
r= 252 =====>(x,y)= (16, 274)
r= 253 =====>(x,y)= (82, 365)
r= 254 =====>(x,y)= (478, 28)
r= 255 =====>(x,y)= (332, 461)
r= 256 =====>(x,y)= (32, 63)
r= 257 =====>(x,y)= (412, 232)
r= 258 =====>(x,y)= (169, 13)
r= 259 =====>(x,y)= (471, 440)
r= 260 =====>(x,y)= (73, 212)
r= 261 =====>(x,y)= (226, 104)
r= 262 =====>(x,y)= (346, 61)
r= 263 =====>(x,y)= (310, 217)
r= 264 =====>(x,y)= (357, 98)
r= 265 =====>(x,y)= (20, 99)
r= 266 =====>(x,y)= (116, 125)
r= 267 =====>(x,y)= (15, 45)
r= 268 =====>(x,y)= (453, 42)
r= 269 =====>(x,y)= (169, 387)
r= 270 =====>(x,y)= (94, 127)
r= 271 =====>(x,y)= (91, 271)
r= 272 =====>(x,y)= (17, 228)
r= 273 =====>(x,y)= (483, 315)
r= 274 =====>(x,y)= (279, 489)
r= 275 =====>(x,y)= (465, 122)
r= 276 =====>(x,y)= (441, 212)
r= 277 =====>(x,y)= (502, 2)
r= 278 =====>(x,y)= (189, 435)
r= 279 =====>(x,y)= (440, 110)
r= 280 =====>(x,y)= (117, 263)
r= 281 =====>(x,y)= (31, 77)
r= 282 =====>(x,y)= (146, 426)
r= 283 =====>(x,y)= (10, 244)
r= 284 =====>(x,y)= (186, 181)
r= 285 =====>(x,y)= (50, 491)
r= 286 =====>(x,y)= (371, 379)
r= 287 =====>(x,y)= (271, 199)
r= 288 =====>(x,y)= (87, 46)
r= 289 =====>(x,y)= (288, 371)
r= 290 =====>(x,y)= (401, 167)
r= 291 =====>(x,y)= (375, 467)
r= 292 =====>(x,y)= (175, 238)
r= 293 =====>(x,y)= (278, 381)
r= 294 =====>(x,y)= (380, 487)
r= 295 =====>(x,y)= (403, 196)
r= 296 =====>(x,y)= (480, 205)
r= 297 =====>(x,y)= (489, 375)
r= 298 =====>(x,y)= (334, 116)
r= 299 =====>(x,y)= (161, 475)
```

```
r= 300 =====>(x,y)= (342, 28)
r= 301 =====>(x,y)= (488, 190)
r= 302 =====>(x,y)= (229, 220)
r= 303 =====>(x,y)= (104, 226)
r= 304 =====>(x,y)= (82, 390)
r= 305 =====>(x,y)= (328, 265)
r= 306 =====>(x,y)= (404, 176)
r= 307 =====>(x,y)= (487, 229)
r= 308 =====>(x,y)= (475, 25)
r= 309 =====>(x,y)= (263, 117)
r= 310 =====>(x,y)= (225, 268)
r= 311 =====>(x,y)= (46, 87)
r= 312 =====>(x,y)= (132, 124)
r= 313 =====>(x,y)= (36, 128)
r= 314 =====>(x,y)= (104, 69)
r= 315 =====>(x,y)= (186, 239)
r= 316 =====>(x,y)= (466, 189)
r= 317 =====>(x,y)= (290, 403)
r= 318 =====>(x,y)= (98, 357)
r= 319 =====>(x,y)= (384, 222)
r= 320 =====>(x,y)= (128, 14)
r= 321 =====>(x,y)= (362, 288)
r= 322 =====>(x,y)= (199, 258)
r= 323 =====>(x,y)= (338, 466)
r= 324 =====>(x,y)= (281, 119)
r= 325 =====>(x,y)= (67, 458)
r= 326 =====>(x,y)= (298, 175)
r= 327 =====>(x,y)= (88, 81)
r= 328 =====>(x,y)= (333, 427)
r= 329 =====>(x,y)= (264, 317)
r= 330 =====>(x,y)= (476, 60)
r= 331 =====>(x,y)= (217, 86)
r= 332 =====>(x,y)= (381, 38)
r= 333 =====>(x,y)= (463, 118)
r= 334 =====>(x,y)= (485, 161)
r= 335 =====>(x,y)= (502, 252)
r= 336 =====>(x,y)= (272, 294)
r= 337 =====>(x,y)= (307, 166)
r= 338 =====>(x,y)= (316, 349)
r= 339 =====>(x,y)= (275, 355)
r= 340 =====>(x,y)= (376, 305)
r= 341 =====>(x,y)= (411, 390)
r= 342 =====>(x,y)= (215, 132)
r= 343 =====>(x,y)= (435, 338)
r= 344 =====>(x,y)= (51, 86)
r= 345 =====>(x,y)= (224, 36)
r= 346 =====>(x,y)= (143, 250)
r= 347 =====>(x,y)= (213, 100)
r= 348 =====>(x,y)= (240, 386)
r= 349 =====>(x,y)= (78, 454)
r= 350 =====>(x,y)= (87, 339)
r= 351 =====>(x,y)= (189, 466)
r= 352 =====>(x,y)= (202, 198)
r= 353 =====>(x,y)= (198, 202)
r= 354 =====>(x,y)= (460, 240)
r= 355 =====>(x,y)= (394, 354)
r= 356 =====>(x,y)= (465, 268)
r= 357 =====>(x,y)= (157, 442)
r= 358 =====>(x,y)= (237, 226)
r= 359 =====>(x,y)= (28, 478)
r= 360 =====>(x,y)= (315, 176)
r= 361 =====>(x,y)= (493, 259)
r= 362 =====>(x,y)= (409, 301)
r= 363 =====>(x,y)= (489, 279)
r= 364 =====>(x,y)= (404, 483)
r= 365 =====>(x,y)= (467, 279)
r= 366 =====>(x,y)= (435, 189)
r= 367 =====>(x,y)= (151, 134)
r= 368 =====>(x,y)= (478, 18)
r= 369 =====>(x,y)= (271, 91)
r= 370 =====>(x,y)= (110, 495)
r= 371 =====>(x,y)= (245, 412)
r= 372 =====>(x,y)= (14, 224)
r= 373 =====>(x,y)= (460, 153)
r= 374 =====>(x,y)= (164, 185)
r= 375 =====>(x,y)= (337, 196)
r= 376 =====>(x,y)= (337, 290)
r= 377 =====>(x,y)= (357, 77)
r= 378 =====>(x,y)= (324, 93)
r= 379 =====>(x,y)= (263, 43)
r= 380 =====>(x,y)= (196, 337)
r= 381 =====>(x,y)= (490, 334)
r= 382 =====>(x,y)= (38, 235)
r= 383 =====>(x,y)= (125, 116)
r= 384 =====>(x,y)= (212, 441)
r= 385 =====>(x,y)= (493, 369)
r= 386 =====>(x,y)= (167, 143)
r= 387 =====>(x,y)= (305, 376)
r= 388 =====>(x,y)= (369, 493)
r= 389 =====>(x,y)= (250, 401)
r= 390 =====>(x,y)= (135, 361)
r= 391 =====>(x,y)= (289, 486)
r= 392 =====>(x,y)= (291, 430)
r= 393 =====>(x,y)= (148, 214)
r= 394 =====>(x,y)= (12, 453)
r= 395 =====>(x,y)= (403, 290)
r= 396 =====>(x,y)= (443, 27)
r= 397 =====>(x,y)= (442, 157)
r= 398 =====>(x,y)= (69, 237)
r= 399 =====>(x,y)= (342, 18)
r= 400 =====>(x,y)= (73, 382)
r= 401 =====>(x,y)= (8, 393)
r= 402 =====>(x,y)= (225, 122)
r= 403 =====>(x,y)= (475, 161)
r= 404 =====>(x,y)= (190, 488)
r= 405 =====>(x,y)= (485, 25)
r= 406 =====>(x,y)= (410, 179)
r= 407 =====>(x,y)= (18, 342)
r= 408 =====>(x,y)= (173, 33)
r= 409 =====>(x,y)= (322, 192)
r= 410 =====>(x,y)= (304, 245)
r= 411 =====>(x,y)= (350, 43)
r= 412 =====>(x,y)= (360, 253)
r= 413 =====>(x,y)= (440, 471)
r= 414 =====>(x,y)= (458, 67)
r= 415 =====>(x,y)= (301, 409)
r= 416 =====>(x,y)= (311, 181)
r= 417 =====>(x,y)= (334, 490)
r= 418 =====>(x,y)= (127, 94)
r= 419 =====>(x,y)= (470, 157)
r= 420 =====>(x,y)= (141, 215)
r= 421 =====>(x,y)= (495, 471)
r= 422 =====>(x,y)= (16, 123)
r= 423 =====>(x,y)= (457, 416)
r= 424 =====>(x,y)= (466, 338)
r= 425 =====>(x,y)= (235, 278)
r= 426 =====>(x,y)= (149, 109)
r= 427 =====>(x,y)= (252, 502)
r= 428 =====>(x,y)= (116, 334)
r= 429 =====>(x,y)= (379, 362)
r= 430 =====>(x,y)= (231, 209)
r= 431 =====>(x,y)= (339, 318)
r= 432 =====>(x,y)= (425, 154)
r= 433 =====>(x,y)= (467, 375)
r= 434 =====>(x,y)= (102, 336)
r= 435 =====>(x,y)= (109, 149)
r= 436 =====>(x,y)= (350, 117)
r= 437 =====>(x,y)= (352, 259)
r= 438 =====>(x,y)= (382, 441)
r= 439 =====>(x,y)= (258, 91)
r= 440 =====>(x,y)= (60, 476)
r= 441 =====>(x,y)= (258, 199)
r= 442 =====>(x,y)= (216, 139)
r= 443 =====>(x,y)= (51, 310)
r= 444 =====>(x,y)= (232, 304)
r= 445 =====>(x,y)= (472, 405)
r= 446 =====>(x,y)= (378, 13)
r= 447 =====>(x,y)= (287, 364)
r= 448 =====>(x,y)= (185, 164)
r= 449 =====>(x,y)= (393, 8)
r= 450 =====>(x,y)= (495, 110)
r= 451 =====>(x,y)= (213, 166)
r= 452 =====>(x,y)= (442, 330)
r= 453 =====>(x,y)= (220, 380)
r= 454 =====>(x,y)= (237, 69)
r= 455 =====>(x,y)= (318, 339)
r= 456 =====>(x,y)= (153, 460)
r= 457 =====>(x,y)= (259, 352)
r= 458 =====>(x,y)= (470, 330)
r=459 is not in the set of definition
r= 460 =====>(x,y)= (86, 51)
r= 461 =====>(x,y)= (63, 32)
r= 462 =====>(x,y)= (416, 457)
r= 463 =====>(x,y)= (378, 387)
r= 464 =====>(x,y)= (124, 141)
r= 465 =====>(x,y)= (338, 435)
r= 466 =====>(x,y)= (490, 125)
r= 467 =====>(x,y)= (382, 73)
r= 468 =====>(x,y)= (43, 350)
r= 469 =====>(x,y)= (31, 98)
r= 470 =====>(x,y)= (436, 313)
r= 471 =====>(x,y)= (187, 49)
r= 472 =====>(x,y)= (171, 12)
r= 473 =====>(x,y)= (217, 310)
r= 474 =====>(x,y)= (170, 76)
r= 475 =====>(x,y)= (318, 46)
r= 476 =====>(x,y)= (461, 50)
r= 477 =====>(x,y)= (40, 422)
r= 478 =====>(x,y)= (121, 62)
r= 479 =====>(x,y)= (368, 142)
r= 480 =====>(x,y)= (176, 404)
r= 481 =====>(x,y)= (386, 153)
r= 482 =====>(x,y)= (25, 485)
r= 483 =====>(x,y)= (480, 265)
r= 484 =====>(x,y)= (411, 365)
r= 485 =====>(x,y)= (327, 188)
r= 486 =====>(x,y)= (283, 123)
r= 487 =====>(x,y)= (352, 369)
r= 488 =====>(x,y)= (77, 31)
r= 489 =====>(x,y)= (315, 483)
r= 490 =====>(x,y)= (328, 205)
r= 491 =====>(x,y)= (330, 470)
r= 492 =====>(x,y)= (125, 490)
r= 493 =====>(x,y)= (415, 385)
r= 494 =====>(x,y)= (491, 332)
r= 495 =====>(x,y)= (311, 239)
r= 496 =====>(x,y)= (42, 171)
r= 497 =====>(x,y)= (23, 298)
r= 498 =====>(x,y)= (307, 100)
r= 499 =====>(x,y)= (212, 73)
r= 500 =====>(x,y)= (283, 274)
r= 501 =====>(x,y)= (386, 240)
r= 502 =====>(x,y)= (441, 382)
```

References

1. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings. In: Gligor, V., Yung, M. (eds.) ACM CCS (2013)
2. Billet, O., Joye, M.: The Jacobi model of an elliptic curve and side-channel analysis. In: Fossorier, M., Høholdt, T., Poli, A. (eds.) AAECC 2003. LNCS, vol. 2643, pp. 34–42. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44828-4_5

3. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
4. Chudnovsky, D.V., Chudnovky, G.V.: Sequences of numbers generated by addition in formal groups and new primality and factorization tests. Adv. Appl. Math. **7**(4), 385–434 (1986)
5. Devigne, J., Joye, M.: Binary huff curves. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 340–355. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19074-2_22
6. Diao, O., Fouotsa, E.: Arithmetic of the level four theta model of elliptic curves. Afrika Mathematica **26**(3), 283–301 (2015)
7. Edwards, H.M.: A normal form for elliptic curves. Bull. Am. Math. Soc. **44**, 393–422 (2007). http://www.ams.org/bull/2007-44-03/S0273-0979-07-01153-6/home.html
8. Diarra, N., Sow, D., Ould Cheikh Khlil, A.Y.: On indifferentiable deterministic hashing into elliptic curves. Eur. J. Pure Appl. Math. **10**(2), 363–391 (2017). (MathScinet: MR3607082) (ZentralBlattMath: Zbl 06701281)
9. Fouotsa, E., Diao, O.: A theta model for elliptic curves. Mediterr. J. Math. **14**, 65 (2017). https://doi.org/10.1007/s00009-017-0840-y
10. Farashahi, R.R.: Hashing into Hessian curves. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 278–289. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21969-6_17
11. Farashahi, R.R., Fouque, P.-A., Shparlinski, I.E., Tibouchi, M., Voloch, J.F.: Indifferentiable deterministic hashing to elliptic and hyperelliptic curves. Math. Comput. **82**(281), 491–512 (2013)
12. Fouque, P.-A., Tibouchi, M.: Estimating the size of the image of deterministic hash functions to elliptic curves. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 81–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14712-8_5
13. Icart, T.: How to hash into elliptic curves. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 303–316. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_18
14. Smart, N.P.: The Hessian form of an elliptic curve. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 118–125. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44709-1_11
15. Shallue, A., van de Woestijne, C.E.: Construction of rational points on elliptic curves over finite fields. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 510–524. Springer, Heidelberg (2006). https://doi.org/10.1007/11792086_36
16. Washington, L.C.: Elliptic Curves. Number Theory and Cryptography. Discrete Mathematics and Applications. Chapman and Hall, Boca Raton (2008)