



Injective Encodings to Binary Ordinary Elliptic Curves

Mojtaba Fadavi¹, Reza Rezaeian Farashahi^{1,2}(✉), and Soheila Sabbaghian¹

¹ Department of Mathematical Sciences, Isfahan University of Technology,
84156-83111 Isfahan, Iran

{mojtaba.fadavi,s.sabbaghian}@math.iut.ac.ir, farashahi@cc.iut.ac.ir

² School of Mathematics, Institute for Research in Fundamental Sciences (IPM),
P.O. Box 19395-5746, Tehran, Iran

Abstract. Representing points of elliptic curves in a way that no pattern can be detected by sensors in the transmitted data is a crucial problem in elliptic curve cryptography. One of the methods that we can represent points of the elliptic curves in a way to be indistinguishable from random bit strings is using injective encoding function. So far, several injective encodings to elliptic curves have been presented, but the previous encoding functions have not supported the binary elliptic curves. More precisely, the only injective encoding to binary elliptic curves was given for Hessian curves, the family of elliptic curves with a point of order 3. In this paper, we propose approaches for constructing injective encoding algorithms to the ordinary binary elliptic curves $y^2 + xy = x^3 + ax^2 + b$ with $\text{Tr}(a) = 1$ as well as those with $\text{Tr}(a + 1) = 0$.

Keywords: Elliptic curve · Cryptography · Injective encoding

2010 Mathematics Subject Classification: 11G05 · 11T06 · 14H52.

1 Introduction

The problem of finding encoding functions from a finite field \mathbb{F}_q into the \mathbb{F}_q -rational points of the given curve was stated by Schoof in 1985 [16]. Such an encoding function is a crucial requirement in the curve-based cryptosystems. For instance, the public key for identity $id \in \{0, 1\}^*$ in the *IBE* scheme, is a \mathbb{F}_q -rational point $Q_{id} = \mathfrak{H}(id)$, where \mathfrak{H} is the desired encoding function. This function is also a requirement for PAKE (Password Authenticated Key Exchange) [5] such as SPEKE (Simple Password Exponential Key Exchange) [13], and PSI (Private Set Intersection) protocols [15].

Bernstein et al. in [3] explained that the traditional methods for encoding to elliptic curves do not disguise the points properly so the encoded points are distinguishable from uniform random bit strings, and consequently, censors can recognize patterns in the transmitted data. To avoid this important drawback, they suggested using a bijection between bit strings and about half of all \mathbb{F}_q -rational points of an elliptic curve E (of j -invariant not equal to 1728) over \mathbb{F}_q

with odd q , where E has a \mathbb{F}_q -rational point of order 2. In the other word, they suggested using injective encoding function, which allows to correspond the set of bit strings $\{0, 1\}^n$ to a subset of $E(\mathbb{F}_q)$. When we use injective encoding function $f : \{0, 1\}^n \rightarrow E(\mathbb{F}_q)$, instead of transferring a point $P \in f(\{0, 1\}^n) \subset E(\mathbb{F}_q)$ we easily transfer the corresponding bit string of P . So far, injective encoding functions are presented for ordinary elliptic curves with non-trivial 3 torsion point by Farashahi [8], non-trivial 4 torsion point by Fouque et al. [9] and non-trivial 2 torsion point by Bernstein et al. [3]. However, for binary ordinary elliptic curves, up to now the only injective encoding function is proposed for binary Hessian elliptic curves [8]. After that, Aranha et al. in [2] using λ -affine coordinate and some computational tricks improved the algorithm in [4] [Appendix E]. But, they did not propose any injective encoding function to binary elliptic curves. To the best of our knowledge, no injective encoding function to ordinary binary elliptic curves has been presented, and this is the main contribution of this paper.

The motivation of this paper is constructing injective encoding function for all ordinary binary elliptic curves, because the previous injective encoding is restricted to binary elliptic curves with a point of order 3 [8]. Two approaches will be proposed in this paper, the first one is applicable to ordinary binary elliptic curves $y^2 + xy = x^3 + ax^2 + b$ where $\text{Tr}(a) = 1$. And, the second is for ordinary binary elliptic curves $y^2 + xy = x^3 + ax^2 + b$ where $\text{Tr}(a+1) = 0$. In fact, ordinary elliptic curves $y^2 + xy = x^3 + ax^2 + b$ over prime extensions of \mathbb{F}_2 with $\text{Tr}(a) = 1$, which are of paramount importance in binary elliptic curve cryptography, belong to both classes at the same time. For instant, all of the five recommended binary elliptic curves by *NIST* have cofactor 2 i.e. the recommended curves are of the form $y^2 + xy = x^3 + ax^2 + b$ with $\text{Tr}(a) = 1$. The proposed algorithms can be applied in protocols which require admissible encoding function to binary elliptic curves. Moreover, since the encoding is injective it behaves as the same as Elligator 2 in [3].

This paper is organized as follows: In Sect. 2, we talk about some different encoding methods from \mathbb{F}_{2^n} to binary elliptic curves and explain what kind of encoding functions are appropriate for cryptography. Besides, we explain an injective function from bit strings to binary finite fields. In Sect. 3, we briefly review the injective encoding function to the binary Hessian curves, and then we will explain our approaches for finding injective encodings to binary elliptic curves.

Throughout the paper, the cardinality of a finite set \mathcal{S} is shown by $\#\mathcal{S}$ and $\|$ denote the concatenation. Also, \mathcal{H} shows a standard hash function.

2 Background

2.1 Elliptic Curves

An elliptic curve is a smooth projective genus 1 curve over a field \mathbb{F} , with a given \mathbb{F} -rational point. Traditionally, an elliptic curve E over a field \mathbb{F} is presented by the Weierstrass equation

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad (1)$$

where the coefficients $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$. Using suitable change of variables, Eq. (1) can be written in the following forms

$$\begin{aligned} E : y^2 &= x^3 + ax^2 + bx + c, & \text{char}(\mathbb{F}) &\neq 2, \\ E : y^2 + xy &= x^3 + ax^2 + b, & \text{char}(\mathbb{F}) &= 2, j(E) \neq 0, \Delta(E) = b, & (2) \\ E : y^2 + cy &= x^3 + ax + b, & \text{char}(\mathbb{F}) &= 2, j(E) = 0, \Delta(E) = a^3, & (3) \end{aligned}$$

where $\Delta(E)$ and $j(E)$ are the discriminant and the j -invariant of the elliptic curve. Elliptic curves in Eqs. (2) and (3) are called ordinary and supersingular binary elliptic curves, respectively. Elliptic curves can be represented by several other models such as Edwards, Hessian, Montgomery, Jacobi intersection, and Jacobi quartic ([1, Chap. 13], [19, Chap. 2], [17]).

The trace function $\text{Tr} : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_2$ is a linear transformation that is defined as follows:

$$\text{Tr}(a) = \sum_{k=0}^{n-1} a^{2^k}.$$

In addition, for n odd, the half trace is the function $\text{HTr} : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$, where

$$\text{HTr}(c) = \sum_{k=0}^{\frac{n-1}{2}} c^{2^{2k}}.$$

An ordinary binary elliptic curve such as E can be transformed into the equation $z^2 + z = g(x)$, where $z = \frac{y}{x}$ and $g(x) = \frac{x^3+ax^2+b}{x^2}$. Consequently, points on E can be found using the solutions of equation $z^2 + z = g(x)$. Also, It is well-known that the equation $z^2 + z = c$ over \mathbb{F}_{2^n} has solution if and only if $\text{Tr}(c) = 0$. And, if z_0 is a solution of this equation then $z_0 + 1$ is one other. More precisely, let $y \in \mathbb{F}_{2^n}$ be an element of trace 1. If $\text{Tr}(c) = 0$ then the solution of equation $z^2 + z = c$ is as follows:

$$Z(c) = \begin{cases} \sum_{k=0}^{\frac{n-3}{2}} c^{2^{2k+1}} & \text{if } n \text{ is odd,} \\ \sum_{k=0}^{n-1} (\sum_{j=0}^k c^{2^j}) y^{2^k} & \text{if } n \text{ is even,} \end{cases} \tag{4}$$

Hence, having an element $y \in \mathbb{F}_{2^n}$, where $\text{Tr}(y) = 1$, we can deterministically compute the roots of quadratic equation $z^2 + z = c$ where $\text{Tr}(c) = 0$.

It is well-known that two ordinary elliptic curves

$$\begin{aligned} E_1 : y^2 + xy &= x^3 + ax^2 + b, & \Delta(E_1) &= b, \\ E_2 : y^2 + xy &= x^3 + \bar{a}x^2 + \bar{b}, & \Delta(E_2) &= \bar{b}, \end{aligned}$$

over \mathbb{F}_{2^n} are isomorphic over \mathbb{F}_{2^n} if and only if $b = \bar{b}$ and $\text{Tr}(a) = \text{Tr}(\bar{a})$ [11]. As a result, we conclude that the number of isomorphism classes of ordinary elliptic curves over \mathbb{F}_{2^n} is $2^{n+1} - 2$. More precisely, fix two elements $\mu, \gamma \in \mathbb{F}_{2^n}$ with $\text{Tr}(\mu) = 0$ and $\text{Tr}(\gamma) = 1$. The set of representative of the isomorphism classes is

$$\mathcal{I} = \mathcal{I}_\mu \cup \mathcal{I}_\gamma,$$

where $\mathcal{I}_a = \{y^2 + xy = x^3 + ax^2 + b \mid b \in \mathbb{F}_{2^n}^*\}$ for $a \in \mathbb{F}_{2^n}$.

Also, any given elliptic curve $E \in \mathcal{I}$ has a non-trivial 2 torsion group, and $\#E(\mathbb{F}_{2^n})$ is divisible by 4 if and only if $E \in \mathcal{I}_\mu$.

Now, we partition \mathcal{I} in terms of n as follows, because we want to investigate different cases separately, according to the requirement of Algorithms 4 and 6.

1. n is odd. Since, $\text{Tr}(1) = 1$ we have

$$\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_0,$$

2. n is even. Since, $\text{Tr}(1) = 0$ we have

$$\mathcal{I} = \mathcal{I}_\gamma \cup \mathcal{I}_1,$$

where $\text{Tr}(\gamma) = 1$.

In Sect. 3, we provide injective encoding algorithms for all ordinary binary elliptic curves $y^2 + xy = x^3 + ax^2 + b$ over \mathbb{F}_{2^n} with $\text{Tr}(a) = 1$ as well as those with $\text{Tr}(a + 1) = 0$.

2.2 Encoding into Elliptic Curves

Encoding into Elliptic Curves: Boneh, et al. in [7] proposed the try-and-increment method. This method is probabilistic, hence it does not run in a constant time so is vulnerable to timing attacks. A variant form of the try-and-increment for elliptic curves over \mathbb{F}_{2^n} with n odd is as follows:

Algorithm 1 Try-and-Increment Algorithm for Ordinary Binary Elliptic Curves

Input: $M \in \{0, 1\}^*$, a random oracle $\mathcal{H} := \{0, 1\}^* \rightarrow \mathbb{F}_{2^n} \times \{0, 1\}$, n odd,
 $E/\mathbb{F}_{2^n} : y^2 + xy = x^3 + ax^2 + b$, and $k \in \mathbb{N}$.
Output: $(x, y) \in E(\mathbb{F}_{2^n})$ or \perp .

```

1:  $i = 0$ ;
2: while  $i < k$  do
3:    $(x, v) = \mathcal{H}(M||i)$ ;  $\triangleright v$  is the least significant bit of  $\mathcal{H}(M||i)$ 
4:    $g(x) = \frac{x^3 + ax^2 + b}{x^2}$ ;
5:   if  $\text{Tr}(g(x)) = 0$  then return  $(x, x(\text{HTr}(g(x)) + v))$ ;
6:   end if
7:    $i = i + 1$ ;
8: end while
return  $\perp$ .
```

The probability of success for any arbitrary $M \in \{0, 1\}^*$ is close to $\frac{1}{2}$. Hence, the probability of failure after up to k rounds is about 2^{-k} , and by taking $k \approx 128$ we are sure that the algorithm will be successful unless in a very rare situations.

Boneh and Franklin in [6] suggested a deterministic method of encoding into elliptic curves, but their method was restricted to the supersingular elliptic

curves of the form $E : y^2 = x^3 + b$ over \mathbb{F}_q , where $q \equiv 2 \pmod{3}$. Although their method was efficient, the MOV attack [14] can be used for transforming the elliptic curve discrete logarithm problem to the finite field version, which has subexponential complexity. They also introduced the following notion of admissible encoding.

Definition 1. ([6]) *A function $f : S \rightarrow R$, where S and R are two finite sets, is an admissible encoding if it satisfies the following conditions:*

- 1: *Computable: f is computable in deterministic polynomial time.*
- 2: *1 to 1: for any $r \in R$, $\#f^{-1}(r) = 1$.*
- 3: *Samplable: there exists a probabilistic polynomial time algorithm that for any $r \in R$, it returns a random element $s \in f^{-1}(r)$.*

Let $f : \mathbb{F}_{2^n} \rightarrow f(\mathbb{F}_{2^n}) \subset E(\mathbb{F}_{2^n})$ be an admissible encoding and $h : \{0, 1\}^* \rightarrow \mathbb{F}_{2^n}$ be a random oracle. Brier et al. in [4] proved that $h : \{0, 1\}^* \rightarrow E(\mathbb{F}_{2^n})$, where $h(m) = f(h(m))$, is indifferentiable from a random oracle to $f(\mathbb{F}_{2^n}) \subseteq E(\mathbb{F}_{2^n})$. So, having an admissible encoding function $f : \mathbb{F}_{2^n} \rightarrow f(\mathbb{F}_{2^n}) \subset E(\mathbb{F}_{2^n})$ is required for having a random oracle into $E(\mathbb{F}_{2^n})$. Also, the inverse function of injective encoding functions can be used in representing points of binary elliptic curves in a way that the preimage is indistinguishable from a uniform bit-string. Now, we review important encoding methods and state their drawbacks.

Icart’s Method: As it was stated above, the Boneh et al.’s encoding function can just be applied to supersingular elliptic curves $y^2 = x^3 + b$. Icart in [12] extended their method and proposed an explicit encoding to elliptic curves $y^2 = x^3 + ax + b$ over \mathbb{F}_q where $q \equiv 2 \pmod{3}$, and ordinary elliptic curves $E : y^2 + xy = x^3 + ax^2 + b$ over \mathbb{F}_{2^n} where n is odd. His binary encoding function is as follows:

$$f_{a,b} : \mathbb{F}_{2^n} \rightarrow E(\mathbb{F}_{2^n})$$

$$u \rightarrow (x, ux + v^2),$$

where $v = a + u + u^2$ and $x = (v^4 + v^3 + b)^{\frac{1}{3}} + v$. In addition, $\#f_{a,b}^{-1}(P) \leq 4$ so it is not 1 : 1, for some small positive integer, and as a result it is not an admissible encoding.

SW Method: Another completely different method that was given before Icart’s method is the Shallue-Woestijne’s method. Their method covers all isomorphism classes of elliptic curves over all finite fields [18], but it was at most 8:1 and as a result it is not an admissible encoding. Here, we recall the binary case of the SW method for ordinary elliptic curves.

Let E be an ordinary binary elliptic curve $y^2 + xy = x^3 + ax^2 + b$, $g(x) = (x^3 + ax^2 + b)/x^2$ and

$$X_1(t, w) = \frac{t(a+w+w^2)}{1+t+t^2}, \quad X_2(t, w) = t.X_1(t, w) + (a + w + w^2),$$

$$X_3(t, w) = \frac{X_1(t,w).X_2(t,w)}{X_1(t,w)+X_2(t,w)}. \tag{5}$$

Then

$$g(X_1(t, w)) + g(X_2(t, w)) + g(X_3(t, w)) \in h(\mathbb{F}_{2^n}),$$

where $h : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$ and $h(z) = z^2 + z$. Since the trace function is a linear transformation on \mathbb{F}_{2^n} , then either one or all of $g(X_i) \in h(\mathbb{F}_{2^n})$. In the other words, we have $\text{Tr}(g(X_i)) = 0$ for either one of $i \in \{1, 2, 3\}$ or all of them. Given such X_i , the solutions of equation $z^2 + z = g(X_i)$ is computable, so the Algorithm 2 always returns 2 or 6 solutions.

Remark 1. We use $Z(c)$ for computing the root of the equation $z^2 + z = c$ in all of the following algorithm. However, for n odd $Z(c)$ is exactly the same as the $\text{HTr}(c)$.

Algorithm 2 is the Shallue-Woestijne algorithm for binary elliptic curves over \mathbb{F}_{2^n} , where n is odd and w is fixed.

Algorithm 2 Binary SW Algorithm

Input: $a, b, t \in \mathbb{F}_{2^n}$, $c = a + w + w^2 \neq 0$, n odd, **and** $E : y^2 + xy = x^3 + ax^2 + b$.
Output: $(x, y) \in E(\mathbb{F}_{2^n})$.

- 1: **if** $t^2 + t + 1 = 0$ **then return** \mathcal{O} ;
 - 2: **end if**
 - 3: $X_1(t) = \frac{tc}{1+t+t^2}$; $X_2(t) = tX_1(t) + c$; $X_3(t) = \frac{X_1(t)X_2(t)}{X_1(t)+X_2(t)}$;
 - 4: **for** $i = 1$ to 3 **do**
 - 5: $g(X_i) = \frac{X_i^3 + aX_i^2 + b}{X_i^2}$;
 - 6: **if** $\text{Tr}(g(X_i)) = 0$ **then return** $(X_i, X_i \text{HTr}(g(X_i)))$;
 - 7: **end if**
 - 8: **end for**
-

The equation of elliptic curve $y^2 + xy = x^3 + ax^2 + b$ in the λ -affine coordinate is of the form $(\lambda^2 + \lambda + a)x^2 = x^4 + b$, where $\lambda = x + \frac{y}{x}$. Aranha et al. in [2] improved Algorithm 2 using the λ -affine coordinate of elliptic curves. More precisely, they fixed t and considered w as the variable parameter and showed that the number of inversions in the Eq. 5 can be decreased to one inversion by using the pre-computed values $\frac{t}{t^2+t+1}, \frac{t+1}{t^2+t+1}, \frac{t^2+t}{t^2+t+1}$. They also used the λ -affine coordinates as a computational trick to eliminate computing inversion to have more efficient binary elliptic curve arithmetic.

2.3 Injective Functions from Bit Strings to \mathbb{F}_{2^n}

To construct an injective encoding function to binary elliptic curves we require an injective encoding from $\{0, 1\}^{n-1}$ to a determined subset S of \mathbb{F}_{2^n} . Here, we explain function κ^l for $l \in \{0, 1\}$ and we use it in Sect. 3.

Let $\Lambda = \{\lambda_1, \dots, \lambda_n\}$ be an arbitrary basis for \mathbb{F}_{2^n} . Then every element $b \in \mathbb{F}_{2^n}$ is uniquely represented by a bit string b_1, b_2, \dots, b_n with $b = \sum_{j=1}^n b_j \lambda_j$.

In particular, $1 = \sum_{j=1}^n c_j \lambda_j$, with $c_i \neq 0$ for a fixed i . Let \mathcal{B}^l , for $l \in \{0, 1\}$, be the subset of \mathbb{F}_{2^n} given by

$$\mathcal{B}^l = \{b : b \in \mathbb{F}_{2^n} \mid b_i = l\}.$$

Now, we can define the function $\kappa^l : \{0, 1\}^{n-1} \rightarrow \mathcal{B}^l \subset \mathbb{F}_{2^n}$, where

$$\kappa^l(b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_n) = \sum_{j=1}^n b_j \lambda_j$$

and $b_i = l$. Clearly, function κ^l is a bijective function and none of the elements of $b = (b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_n) \in \{0, 1\}^{n-1}$ is sent to $\sum_{j=1}^n (b_j + c_j) \lambda_j = 1 + \sum_{j=1}^n b_j \lambda_j$. As a result, for any $w \in \mathbb{F}_{2^n}$ one and only one of w or $w + 1$ belongs to \mathcal{B}^l .

For example, if $\Lambda = \{1, \alpha, \alpha^2, \dots, \alpha^{n-1}\}$ is the polynomial basis of \mathbb{F}_{2^n} , then \mathcal{B}^l is the set of elements in \mathbb{F}_{2^n} with the least significant bit l , where $l \in \{0, 1\}$. Now, we can define the bijective functions $\kappa^l : \{0, 1\}^{n-1} \rightarrow \mathcal{B}^l \subset \mathbb{F}_{2^n}$ for $l \in \{0, 1\}$, where

$$\kappa^l(b_2, b_3, \dots, b_n) = l + \sum_{j=2}^n b_j \alpha^{j-1}.$$

Hereafter, we let $\kappa = \kappa^0$.

3 Injective Encoding to Binary Elliptic Curves

In this section, we first recall the injective encoding function to binary elliptic curves with a point of order 3 [8]. Then, we present two Algorithms which bring about injective encoding for all ordinary binary elliptic curves $E : y^2 + xy = x^3 + ax^2 + b$ with $\text{Tr}(a) = 1$ or $\text{Tr}(a + 1) = 0$, respectively.

3.1 Encoding into Hessian Curves

Up to now, the only injective encoding to binary elliptic curves has been given for the Hessian form of elliptic curves over \mathbb{F}_{2^n} with n odd [8]. A binary Hessian elliptic curve has a point of order 3, therefore that injective encoding is applicable only to the family of binary elliptic curves with a point of order 3. More precisely, let

$$H_d : x^3 + y^3 + 1 = dxy,$$

where $d \in \mathbb{F}_{2^n}$ and $d^3 \neq 1$, be an Hessian curve over a finite field \mathbb{F}_{2^n} with n odd [10]. It is shown in [8] that there is an injective function $\text{e1t} : \{0, 1\}^{n-1} \rightarrow \mathbb{F}_{2^n}$ in which $\text{Tr}(d^3(\text{e1t}(b)^2 + \text{e1t}(b))) = 0$, for all $b \in \{0, 1\}^{n-1}$. Therefore, the following map is well defined and injective.

$$i_d : \{0, 1\}^{n-1} \longrightarrow H_d(\mathbb{F}_{2^n})$$

where $i_d(b) = (x, y)$ if $\text{elt}(b) \neq 0$, and $x = duv$, $y = d(u + v)$ with

$$u = \frac{1}{d} \left(\frac{w}{\text{elt}(b)} \right)^{1/3}, \quad v = \frac{1}{d} \left(\frac{w+1}{\text{elt}(b)} \right)^{1/3},$$

$$w^2 + w = d^3(\text{elt}(b)^2 + \text{elt}(b)),$$

and $i_d(b) = (1, 0)$ if $\text{elt}(b) = 0$.

3.2 Injective Encoding to Binary Elliptic Curves with $\text{Tr}(a) = 1$.

Let E be the following ordinary binary elliptic curve

$$E : y^2 + xy = x^3 + ax^2 + b, \quad \text{Tr}(a) = 1. \tag{6}$$

Here, we explain our first approach for finding injective encoding function from $\{0, 1\}^{n-1}$ to elliptic curves with Eq. (6).

As we recall, Eq. (5) is a two variables function in w and t . The main idea for finding a new injective encoding from $\{0, 1\}^{n-1}$ to the ordinary binary elliptic curves, is fixing t and going through all $w \in \mathbb{F}_{2^n}$. However, to achieve such injective encoding we require to have $\text{Tr}(a) = 1$, and binary elliptic curves which are used in elliptic curve cryptography are exactly ordinary binary elliptic curves with $\text{Tr}(a) = 1$. SW algorithm for binary elliptic curves $y^2 + xy = x^3 + ax^2 + b$, when we fix $t \in \mathbb{F}_{2^n}$ and consider $w \in \mathbb{F}_{2^n}$ as a variable, is the following algorithm and we use the notation f for Algorithm 3 to call it in Algorithm 4.

Algorithm 3 Encoding to Binary Elliptic Curves $y^2 + xy = x^3 + ax^2 + b$.

Input: $w, a, b \in \mathbb{F}_{2^n}$, **where** $t(t+1)(t^2+t+1) \neq 0$, $s = \frac{t}{t^2+t+1}$, $r = \frac{t+1}{t}$,

and $E/\mathbb{F}_{2^n} : y^2 + xy = x^3 + ax^2 + b$.

Output: $(x, y) \in E(\mathbb{F}_{2^n})$.

- 1: $c = a + w + w^2$;
 - 2: **if** $c(1+c) = 0$ **then**, Return \mathcal{O} ;
 - 3: **end if**
 - 4: $X_1 = sc$; $X_2 = rX_1$; $X_3 = trX_1$;
 - 5: **for** $i = 1$ to 3 **do**
 - 6: $g(X_i) = \frac{X_i^3 + aX_i^2 + b}{X_i^2}$;
 - 7: **if** $\text{Tr}(g(X_i)) = 0$ **then** $x = X_i$; $y = X_i \cdot Z(g(X_i))$;
 - 8: **end if**
 - 9: **end for**
 - return** (x, y) .
-

Remark 2. Clearly, $f(w) = P$ if and only if $f(w+1) = P$, so for a given point $P \in f(\mathbb{F}_{2^n})$, $\mathcal{V}_P = \{w_1, w_1+1, w_2, w_2+1, w_3, w_3+1\} \subset \mathbb{F}_{2^n}$ is the largest possible preimage set of P and by considering the set $\mathcal{W}_P = \{w_1, w_2, w_3\} \subset \mathcal{V}_P$ as the preimage set of P , we do not lose information about the preimages of P .

The following proposition shows that there is an interesting feature in Algorithm 3 which can be used for providing a 2:1 encoding from \mathbb{F}_{2^n} to binary elliptic curves of Eq. (6).

Proposition 3. *If $\text{Tr}(a) = 1$, then Algorithm 3 is at most 4 : 1.*

Proof. Since $\text{Tr}(a) = 1$ we conclude that at most two elements of \mathbb{F}_{2^n} are sent to \mathcal{O} . Now, suppose that we are given a point $P = (x_0, y_0) \in \mathfrak{f}(\mathbb{F}_{2^n}) \subset E(\mathbb{F}_{2^n})$. We consider two possibilities.

1. If $X_1 = \frac{t}{t^2+t+1}(a + w + w^2) \neq x_0$ for all $w \in \mathbb{F}_{2^n}$. Clearly, $\#\mathfrak{f}^{-1}(P) \leq 4$ because $\deg(X_2) = \deg(X_3) = 2$, and we are done.
2. If $X_1(w_1) = X_1(w_1 + 1) = \frac{t}{t^2+t+1}(a + w_1 + w_1^2) = x_0$. In this case, it is impossible that we have $X_2(w_2) = \frac{t+1}{t^2+t+1}(w_2^2 + w_2 + a) = x_0$ and $X_3(w_3) = \frac{t(t+1)}{t^2+t+1}(w_3^2 + w_3 + a) = x_0$ simultaneously.

Because, if this happens then $\mathfrak{f}^{-1}(P) = \{w_1, w_1 + 1, w_2, w_2 + 1, w_3, w_3 + 1\} \subset \mathbb{F}_{2^n}$ and we have

$$x_0 = \frac{t}{t^2 + t + 1}(w_1^2 + w_1 + a) = \frac{t + 1}{t^2 + t + 1}(w_2^2 + w_2 + a), \tag{7}$$

$$x_0 = \frac{t}{t^2 + t + 1}(w_1^2 + w_1 + a) = \frac{t(t + 1)}{t^2 + t + 1}(w_3^2 + w_3 + a), \tag{8}$$

or equivalently

$$w_2^2 + w_2 + \left(\frac{t(w_1^2 + w_1) + a}{t + 1}\right) = 0, \tag{9}$$

$$w_3^2 + w_3 + \left(\frac{w_1^2 + w_1 + ta}{t + 1}\right) = 0. \tag{10}$$

Now, if we let $A = \frac{t(w_1^2+w_1)+a}{t+1}$ and $B = \frac{w_1^2+w_1+ta}{t+1}$, we see that $A + B = w_1^2 + w_1 + a$ and $\text{Tr}(A + B) = \text{Tr}(w_1^2 + w_1 + a) = \text{Tr}(a) = 1$. Therefore, we conclude that one and only one of the Eqs. (9) or (10) has solution. Hence, one of the Eqs. (7) or (8) is held and $\#\mathfrak{f}^{-1}(P) = 4$. □

Now, let we are given a point $P \in \mathfrak{f}(\mathbb{F}_{2^n})$. Since P has at most four preimages, we have to first modify Algorithm 3 to have a 2:1 encoding function then using the bijective function in Sect. 2.3 we can construct our desired injective encoding from $\{0, 1\}^{n-1}$ to $E(\mathbb{F}_{2^n})$.

Theorem 4. *Let E be the elliptic curve of Eq. (6). There is a function $\mathfrak{g} : \mathbb{F}_{2^n} \rightarrow E(\mathbb{F}_{2^n})$ which is 2:1. In addition, \mathfrak{g}^{-1} is computable.*

Proof. Since $\text{Tr}(a) = 1$, by Proposition 3 we conclude that Algorithm 3 is at most 4 : 1. So, we have two main possibilities for the preimage set of any point $P = (x_0, y_0) \in \mathfrak{f}(\mathbb{F}_{2^n})$.

1. $\#\mathfrak{f}^{-1}(P) = 4$. Let $\mathcal{W}_P = \{w, \lambda\}$ be the preimage set of $\mathfrak{f}^{-1}(P)$, then $\mathcal{W}_P = \{w_i, w_j\}$, where $i, j \in \{1, 2, 3\}$ and $i \neq j$. Also, the index i of w_i refers to the index of X_i which produces point P . So, we have the following three cases.

(a) $\{w, \lambda\} = \{w_1, w_2\}$. Let R_1 and R_{31} be the sets of roots of equations

$$x^2 + x + \left(\frac{t(w^2 + w) + a}{t + 1}\right) = 0,$$

$$x^2 + x + \frac{(t + 1)(w^2 + w) + a}{t} = 0.$$

These two equations are related to each other, in the sense that if we simplify the first equation regarding to w and swap x with w , we get the other equation and vice versa. Now, there are two cases

- i. If $w = w_1$ and $\lambda = w_2$ then $R_1 \subseteq \{\lambda, \lambda + 1\}$ and $R_{31} \subseteq \{\zeta, \zeta + 1\}$.
 - ii. If $w = w_2$ and $\lambda = w_1$ then $R_1 \subseteq \{\zeta, \zeta + 1\}$ and $R_{31} \subseteq \{w, w + 1\}$, where $\{\zeta, \zeta + 1\} \cap f^{-1}(P) = \emptyset$. In each case, we can use the function f of Algorithm 3 to investigate which set is the suitable set. For the first case, we define $\mathbf{g}(w) = f(w)$, and for the second case we define $\mathbf{g}(w) = -f(w)$.
- (b) $\{w, \lambda\} = \{w_1, w_3\}$. Let R_2 and R_{32} be the sets of roots of equations

$$x^2 + x + \frac{w^2 + w + ta}{t + 1} = 0,$$

$$x^2 + x + (t + 1)(w^2 + w) + ta = 0.$$

- i. If $w = w_1$ and $\lambda = w_3$ then $R_2 \subseteq \{\lambda, \lambda + 1\}$ and $R_{32} \subseteq \{\zeta, \zeta + 1\}$.
 - ii. If $w = w_3$ and $\lambda = w_1$ then $R_2 \subseteq \{\zeta, \zeta + 1\}$ and $R_{32} \subseteq \{w, w + 1\}$, where again $\{\zeta, \zeta + 1\} \cap f^{-1}(P) = \emptyset$. Similar to the first case, we use the function f to investigate which set is the suitable set. For the first case, we define $\mathbf{g}(w) = f(w)$, and for the second case we define $\mathbf{g}(w) = -f(w)$.
- (c) $\{w, \lambda\} = \{w_2, w_3\}$. Let R_4 and R_5 be the sets of roots of equations

$$x^2 + x + \frac{w^2 + w + (t + 1)a}{t} = 0,$$

$$x^2 + x + t(w^2 + w) + (t + 1)a = 0.$$

- i. If $w = w_2$ and $\lambda = w_3$ then $R_4 \subseteq \{\lambda, \lambda + 1\}$ and $R_5 \subseteq \{\zeta, \zeta + 1\}$.
- ii. If $w = w_3$ and $\lambda = w_2$ then $R_4 \subseteq \{\zeta, \zeta + 1\}$ and $R_5 \subseteq \{w, w + 1\}$.

Like the previous cases, for the first case we define $\mathbf{g}(w) = f(w)$, and for the second case we define $\mathbf{g}(w) = -f(w)$.

2. $\#f^{-1}(P) = 2$. Let $\mathcal{W}_P = \{w\}$.

In this case, none of the sets of $R_1, R_2, R_{31}, R_{32}, R_4$ and R_5 are allowed to output. So, we just define $\mathbf{g}(w) = f(w)$.

For computing $\mathbf{g}^{-1}(P)$, where $P = (x, y) \in \mathbf{g}(\mathbb{F}_{2^n})$, we consider the list

$$L = [a + x(t + 1 + s_1), a + x(t + s_2), a + x(1 + s_1 s_2)],$$

where $s_1 = \frac{1}{t}, s_2 = \frac{1}{t+1}$ and since t is fixed we only use the precomputed value of s_1 and s_2 . The preimage of P is the element $l \in L$ which satisfies the necessary property $\text{Tr}(l) = 0$. For such $l \in L$, we accept the solution $w_0 = Z(l)$ of the equation $w^2 + w + l = 0$ as the desired preimage if $\mathbf{g}(w_0) = P$. □

Algorithm 4 Encoding to Binary Elliptic curves $y^2 + xy = x^3 + ax^2 + b$ with $\text{Tr}(a) = 1$.

Input: $w, a, b \in \mathbb{F}_{2^n}$, **where** $\text{Tr}(a) = 1, t(t+1)(t^2+t+1) \neq 0, s = \frac{t}{t^2+t+1},$
 $r = \frac{t+1}{t}, s_1 = \frac{1}{t}, s_2 = \frac{1}{t+1}$ **and** $E : y^2 + xy = x^3 + ax^2 + b.$
Output: $(x, y) \in E(\mathbb{F}_{2^n}).$

```

1: if  $(1 + a + w + w^2) = 0$  then, return  $\mathcal{O}$ ;
2: end if
3:  $L = [(t(w^2 + w) + a)s_2, ((t + 1)(w^2 + w) + a)s_1, (w^2 + w + ta)s_2, (t + 1)(w^2 + w) +$ 
 $ta, (w^2 + w + (t + 1)a)s_1, t(w^2 + w) + (t + 1)a];$ 
4: for  $i = 1$  to 6 do
5:   if  $\text{Tr}(L[i]) = 0$  then  $z = Z(L[i]);$ 
6:     if  $f(w) = f(z)$  then return  $(-1)^{i+1}f(w);$ 
7:     end if
8:   end if
9: end for
return  $f(w);$ 

```

Algorithm 5 explains details of computing the preimage of a point $P \in \mathfrak{g}(\mathbb{F}_{2^n})$. It should be mentioned that we don't require the exact preimage of a point P . In fact, since $\mathfrak{g}^{-1}(P) = \{w, w + 1\}$ we are able to find another preimage of P using output of Algorithm 5 and that is sufficient for constructing our injective encoding function.

The following proposition describes how we can extract an injective encoding function by composing the functions \mathfrak{g} and $\kappa : \{0, 1\}^{n-1} \rightarrow \mathbb{F}_{2^n}$.

Proposition 5. *Function $\mathfrak{g} \circ \kappa : \{0, 1\}^{n-1} \rightarrow E(\mathbb{F}_{2^n})$ is an injective encoding function.*

Proof. Function \mathfrak{g} is 2:1 with this property that, for all $w \in \mathbb{F}_{2^n}$, $\mathfrak{g}(w) = \mathfrak{g}(w + 1)$. On the other hand, the injective function $\kappa : \{0, 1\}^{n-1} \rightarrow \mathbb{F}_{2^n}$ covers one and only one of the elements w or $w + 1$. Therefore, function $\mathfrak{g} \circ \kappa : \{0, 1\}^{n-1} \rightarrow E(\mathbb{F}_{2^n})$ will be the desired injective encoding function. □

Algorithm 5 Computing the preimage of $P \in \mathfrak{g}(\mathbb{F}_{2^n})$.

Input: $E : y^2 + xy = x^3 + ax^2 + b$, **where** $a, b \in \mathbb{F}_{2^n}, \text{Tr}(a) = 1, t(t+1)(t^2+t+1) \neq 0,$
 $s_1 = \frac{1}{t}, s_2 = \frac{1}{t+1}$ **and** $P \in E(\mathbb{F}_{2^n}).$
Output: $w \in \mathbb{F}_{2^n}$, **where** $\mathfrak{g}(w) = P$, **or** $\emptyset.$

```

1: if  $P = \mathcal{O}$  then, return  $w = Z(a + 1);$ 
2: end if
3:  $L = [a + x(t + 1 + s_1), a + x(t + s_2), a + x(1 + s_1 s_2)];$ 
4: for  $i = 1$  to 3 do
5:   if  $\text{Tr}(L[i]) = 0$  then  $w = Z(L[i]);$ 
6:     if  $\mathfrak{g}(w) = P$  then return  $w;$ 
7:     end if
8:   end if
9: end for
return  $\emptyset$ 

```

3.3 Injective Encoding to Binary Elliptic Curves with $\text{Tr}(a + 1) = 0$

Here, we describe our second simple approach for finding an injective encoding to the family of binary elliptic curves

$$E : y^2 + xy = x^3 + ax^2 + b, \quad \text{Tr}(a + 1) = 0. \tag{11}$$

We remark that, this method can be seen as simplified SW algorithm.

Proposition 6. *Let E be an elliptic curve over \mathbb{F}_{2^n} given by the Eq. (11). Then, for every $t \in \mathbb{F}_{2^n}$ there exists a point P on E with $x(P)$ equals $t, t + 1$ or $t^2 + t$.*

Proof. For t where $t^2 + t = 0$ we have the point $P = (0, \sqrt{b})$. Now, let $g(x) = x + a + \frac{b}{x^2}$. Then, for all $t \in \mathbb{F}_{2^n}^*$, we have $g(t) + g(t + 1) + g(t^2 + t) = t^2 + t + a + 1$. Using the linearity of Trace function, we have

$$\text{Tr}(g(t)) + \text{Tr}(g(t + 1)) + \text{Tr}(g(t^2 + t)) = \text{Tr}(t^2 + t + a + 1) = 0.$$

So, there exist a point P on E where $x(P) \in \{t, t + 1, t^2 + t\}$ and $y(P) = x(P)Z(g(x(P)))$ (see Eq. 4). □

The trivial solution to correspond an element $t \in \mathbb{F}_{2^n}$ to a point on binary elliptic curve E , is to check whether there is a point with x -coordinate equals t or $t + 1$. But, what about the case if it fails? For the family of elliptic curves E of the form (11), Proposition 6 shows there is a point on E with x -coordinate equals $t^2 + t$ if there is no points with x -coordinate equal to t and $t + 1$. To make this encoding uniform 2:1, the first step is to find a point on E for the value $t^2 + t$ and if it fails the second step is for the values t and $t + 1$. Here the output of encoding is the same for input values t and $t + 1$. Also, the main technical point is using the negation map on E to make a distinction between these two steps.

Now, we present Algorithm 6 which is 2:1 from \mathbb{F}_{2^n} to $E(\mathbb{F}_{2^n})$, where E is the elliptic curve with Eq. (11). Using Algorithm 6 we can construct our desired injective encoding from $\{0, 1\}^{n-1}$ to $E(\mathbb{F}_{2^n})$.

Algorithm 6 Encoding to Binary Elliptic Curves $y^2 + xy = x^3 + ax^2 + b$ with $\text{Tr}(a + 1) = 0$.

Input: $t, a, b \in \mathbb{F}_{2^n}$, $E : y^2 + xy = x^3 + ax^2 + b$, with $\text{Tr}(a + 1) = 0$.

Output: $(x, y) \in E(\mathbb{F}_{2^n})$.

- 1: $X_1 = t; \quad X_2 = t + 1; \quad X_3 = t^2 + t;$
- 2: **if** $X_3 = 0$ **then return** $(0, \sqrt{b});$
- 3: **end if**
- 4: $g_1 = X_1 + a + \frac{b}{X_1^2}; \quad g_2 = X_2 + a + \frac{b}{X_2^2}; \quad g_3 = X_3 + a + \frac{b}{X_3^2};$
- 5: **if** $\text{Tr}(g_3) = 0$ **then return** $(X_3, X_3(Z(g_3) + 1));$
- 6: **else**
- 7: **if** $\text{Tr}(g_1) = 0$ **then return** $(X_1, X_1Z(g_1));$
- 8: **else return** $(X_2, X_2Z(g_2));$
- 9: **end if**
- 10: **end if**

Theorem 7. *Function $\epsilon : \mathbb{F}_{2^n} \rightarrow E(\mathbb{F}_{2^n})$ given by Algorithm 6 is 2:1. Furthermore, $\epsilon^{-1}(P)$ is computable.*

Proof. Let $P = (u, v)$ be an affine point of $E(\mathbb{F}_{2^n})$. Clearly $P = (u, v) \in \epsilon(\mathbb{F}_{2^n})$ only if there exists some $t \in \mathbb{F}_{2^n}$ such that u equals $t, t + 1$ or $t^2 + t$. In other words $\epsilon^{-1}(P) \subset \{u, u + 1, w, w + 1\}$, where $w \in \mathbb{F}_{2^n}$ and $w^2 + w = u$. Obviously, to compute $\epsilon^{-1}(P)$, we find elements $t \in \{u, u + 1, w, w + 1\}$ that is mapped to P by ϵ .

If $u = 0$ then $P = (0, \sqrt{b})$. Clearly, $\epsilon^{-1}(P) = \{0, 1\}$. From now on, we assume $u \neq 0$. For $x \in \mathbb{F}_{2^n}^*$, let $g(x) = x + a + b/x^2$ and $T(x) = \text{Tr}(g(x))$. Clearly there exists a point $P = (u, v)$ on E if and only if $T(u) = 0$. Then we have, $v = uZ(g(u))$ or $v = u(Z(g(u)) + 1)$ (see Sect. 2.1). For the point $P = (u, v)$ on E with $u \neq 0$, let $c(P) = v/u + Z(g(u))$. Then, $P = (u, u(Z(g(u)) + c(P)))$. Clearly the compression of point P or $-P$ is given by $x(P) = x(-P) = u$ and the bit $c(P)$ or $1 + c(P)$ respectively.

We consider the following cases for u .

1. Let u be such that $\text{Tr}(u) = 0$, then let fix $w \in \mathbb{F}_{2^n}$ such that $w^2 + w = u$. Then

$$\epsilon^{-1}(P) \subset \{u, u + 1, w, w + 1\}.$$

For the point P with $u = 1$, we have $\epsilon^{-1}(P) = \{w, w + 1\}$ if $c(P) = 1$ and $\epsilon^{-1}(P) = \emptyset$ otherwise. Now, we assume $u \neq 1$. From Proposition 6, we have

$$T(u) + T(u + 1) + T(u^2 + u) = 0, \quad T(w) + T(w + 1) + T(u) = 0.$$

Since $T(u) = 0$, there are 4 possibilities for the values $T(w), T(w + 1), T(u + 1)$ and $T(u^2 + u)$.

From Algorithm 6, we check the output of ϵ for following cases of the input t .

- For all $t \in \{u, u + 1\}$, if $T(u^2 + u) = 0$ we have $x(\epsilon(t)) = u^2 + u \neq u$, since $u \neq 0$, so $\epsilon(t) \neq \pm P$. Also, if $T(u^2 + u) = 1$, we have $\epsilon(t) = P$ if $c(P) = 0$ and $\epsilon(t) = -P$ if $c(P) = 1$.
- For all $t \in \{w, w + 1\}$, we have $x(\epsilon(t)) = w^2 + w = u$. Then $y(\epsilon(t)) = u(Z(g(u)) + 1) \neq y(P)$ if $c(P) = 0$, and $y(\epsilon(t)) = u(Z(g(u)) + 1) = y(P)$ if $c(P) = 1$. In other words, for all $t \in \{w, w + 1\}$, we have $\epsilon(t) = -P$ if $c(P) = 0$ and $\epsilon(t) = P$ if $c(P) = 1$.

Then, we compute $\epsilon^{-1}(P)$ for all possible cases of $c(P)$ and $T(u^2 + u)$.

- If $c(P) = 0$ and $T(u^2 + u) = 0$, then we have $\epsilon^{-1}(P) = \emptyset$.
 - If $c(P) = 0$ and $T(u^2 + u) = 1$, then we see that $\epsilon^{-1}(P) = \{u, u + 1\}$.
 - If $c(P) = 1$ then $\epsilon^{-1}(P) = \{w, w + 1\}$.
2. If $\text{Tr}(u) = 1$, then there is no element $w \in \mathbb{F}_{2^n}$ such that $w^2 + w = u$. So,

$$\epsilon^{-1}(P) \subset \{u, u + 1\}.$$

Similar to the previous case, we have $T(u) + T(u + 1) + T(u^2 + u) = 0$. Also, $\epsilon^{-1}(P) = \{u, u + 1\}$ if $T(u^2 + u) = 1$ and $c(P) = 0$ and $\epsilon^{-1}(P) = \emptyset$ otherwise.

Briefly, for all $P = (u, v) \in \mathfrak{e}(\mathbb{F}_{2^n})$, we have

$$\mathfrak{e}^{-1}(P) = \begin{cases} \{u, u + 1\} & \text{if } u = 0, \\ \{u, u + 1\} & \text{if } c(P) = 0 \text{ and } \text{Tr}(a + \frac{b}{u^4 + u^2}) = 1, \ u \neq 0, 1, \\ \{w, w + 1\} & \text{if } c(P) = 1 \text{ and } \text{Tr}(u) = 0, \\ \emptyset & \text{otherwise.} \end{cases}$$

Hence, the function \mathfrak{e} is 2:1. □

Algorithm 7 describes computing preimage of a point $P \in \mathfrak{e}(\mathbb{F}_{2^n})$.

Proposition 8. *Function $\mathfrak{e} \circ \kappa : \{0, 1\}^{n-1} \rightarrow E(\mathbb{F}_{2^n})$, $l \in \{0, 1\}$, is an injective encoding function.*

Proof. The proof line is the same as Proposition 5. □

Note that Algorithm 7 for a given point $P \in E(\mathbb{F}_{2^n})$ outputs an element t in \mathbb{F}_{2^n} or gives nothing. Notice, t is represented by a bit string of length n . For computing the preimage of P by $\mathfrak{e} \circ \kappa$, the required output is a bit sting of length $n - 1$, where simply is obtained by removing a single bit of t in the suitable fixed position. More precisely, for the basis $\Lambda = \{\lambda_1, \dots, \lambda_n\}$ of \mathbb{F}_{2^n} , let fix i such that $1 = \sum_{j=1}^n c_j \lambda_j$, with $c_i \neq 0$. From Sect. 2.3, we recall the injective functions $\kappa^l : \{0, 1\}^{n-1} \rightarrow \mathbb{F}_{2^n}$, for $l = 0, 1$. The preimage of $t = \sum_{j=1}^n t_j \lambda_j$ by one of these functions is the required output bit string $(t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_n)$ in $\{0, 1\}^{n-1}$.

Algorithm 7 Computing the preimage of $P \in E(\mathbb{F}_{2^n})$.

Input: $E : y^2 + xy = x^3 + ax^2 + b$, **where** $a, b \in \mathbb{F}_{2^n}$, $\text{Tr}(a + 1) = 0$, **and** $P = (x, y) \in E(\mathbb{F}_{2^n})$.

Output: $t \in \mathbb{F}_{2^n}$, **where** $\mathfrak{e}(t) = P$, **or** \emptyset .

- 1: $u = x(P)$;
 - 2: **if** $u = 0$ **then return** 0;
 - 3: **end if**
 - 4: $v = y(P)$;
 - 5: $T_u = \text{Tr}(u)$;
 - 6: **if** $T_u = 0$ **then**, $w = Z(u)$;
 - 7: **end if**
 - 8: **if** $\frac{v}{u} = Z(u + a + b/u^2)$ **then** $c_P = 0$;
 - 9: **else** $c_P = 1$;
 - 10: **end if**
 - 11: **if** $u = 1$ **then** $T = 0$;
 - 12: **else** $T = \text{Tr}(a + \frac{b}{u^4 + u^2})$;
 - 13: **end if**
 - 14: **if** $c_P = 0$ **and** $T = 1$ **then return** u ;
 - 15: **end if**
 - 16: **if** $c_P = 1$ **and** $T_u = 0$ **then return** w ;
 - 17: **end if**
 - return** \emptyset
-

4 Concluding Remarks

It is well-known that the encoding functions from \mathbb{F}_{2^n} to the binary elliptic curves are non-uniform. In fact, the SW-method and the Icart's method, are at most 6:1 and 4:1, respectively. But, we require to have uniform encoding function, because the transmitted data have to be indistinguishable from the uniform bit strings. In this regard, we can use the injective encoding function to the binary elliptic curves as an admissible encoding. So far, the only injective encoding function to binary elliptic curves is given for those with a point of order 3. In this paper, we studied the general case of binary elliptic curves, and we proposed encoding algorithms which provide us injective encoding functions into binary elliptic curves. Algorithms 4 and 6 covers elliptic curves with equation $y^2 + xy = x^3 + ax^2 + b$ with $\text{Tr}(a) = 1$ and $\text{Tr}(a + 1) = 0$, respectively. These algorithms are both 2:1 and the preimage of a point P in the image of functions is $\{w, w + 1\}$, for some $w \in \mathbb{F}_{2^n}$. So using a suitable injective function $\kappa : \{0, 1\}^{n-1} \rightarrow \mathbb{F}_{2^n}$, which covers one and only one of the elements of the set $\{w, w + 1\}$, we construct injective encoding function from $\{0, 1\}^{n-1}$ to the given elliptic curves.

Acknowledgment. The authors thank Diego Aranha and Anonymous reviewers for the useful comments of this work. This research was in part supported by a grant from IPM (No. 96050416).

References

1. Avanzi, R., et al.: Handbook of Elliptic and Hyperelliptic Curve Cryptography. CRC Press, Boca Raton (2005)
2. Aranha, D.F., Fouque, P.-A., Qian, C., Tibouchi, M., Zapolowicz, J.-C.: Binary elligator squared. In: Joux, A., Youssef, A. (eds.) SAC 2014. LNCS, vol. 8781, pp. 20–37. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13051-4_2
3. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM Conference on Computer and Communications Security, pp. 967–980. ACM (2013)
4. Brier, E., Coron, J.-S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient indifferentiable hashing into ordinary elliptic curves. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 237–254. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_13
5. Boyko, V., MacKenzie, P., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_12
6. 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
7. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_30

8. 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
9. Fouque, P.-A., Joux, A., Tibouchi, M.: Injective encodings to elliptic curves. In: Boyd, C., Simpson, L. (eds.) ACISP 2013. LNCS, vol. 7959, pp. 203–218. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39059-3_14
10. Hesse, O.: Über die Elimination der Variabeln aus drei algebraischen Gleichungen vom zweiten Grade mit zwei Variabeln. *J. Reine Angew. Math.* **10**, 68–96 (1844)
11. Hankerson, D., Menezes, A.J., Vanstone, S.: *Guide to Elliptic Curve Cryptography*, 1st edn. Springer, New York (2004). <https://doi.org/10.1007/b97644>
12. 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
13. Jablon, D.P.: Strong password-only authenticated key exchange. *SIGCOMM Comput. Commun.* **26**(5), 5–26 (1996)
14. Menezes, A., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field, pp. 1639–1647. *IEEE* (1993)
15. Resende, A.C.D., Aranha, D.F.: Faster unbalanced private set intersection. *J. Internet Serv. Appl.* **9**(1), 1–18 (2018)
16. Schoof, R.: Elliptic curves over finite fields and the computation of square roots mod p . *Math. Comput.* **44**(170), 483–494 (1985)
17. Silverman, J.H.: *The Arithmetic of Elliptic Curves*. Springer, Berlin (1995)
18. 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
19. Washington, L.C.: *Elliptic Curves: Number Theory and Cryptography*, 2nd edn. CRC Press, Boca Raton (2008)