On τ-adic Representations of Integers

N. Ebeid and M. A. Hasan

Department of Electrical and Computer Engineering

and

Centre for Applied Cryptographic Research

University of Waterloo

Ontario, Canada

June 11, 2007

Abstract

Elliptic curve cryptosystems have become increasingly popular due to their efficiency and the small size of the keys they use. Particularly, the anomalous curves introduced by Koblitz allow a complex representation of the keys, denoted τNAF, that make the computations over these curves more efficient. In this article, we propose an efficient method for randomizing a τNAF to produce different equivalent representations of the same key to the same complex base τ. We prove that the average Hamming density of the resulting representations is 0.5. We identify the pattern of the τNAFs yielding the maximum number of representations and the formula governing this number. We also present deterministic methods to compute the average and the exact number of possible representations of a τNAF.

1 Introduction

Elliptic curve cryptosystems (ECCs) have become increasingly popular due to the efficiency of their computations and the small size of their keys compared to RSA and discrete logarithm-based systems. They rely on the hardness of solving the discrete logarithm problem (DLP) in the additive group of points on the elliptic curve E defined over a finite field $\mathbb{F}_q$. The core and most costly operation in ECCs is the *scalar multiplication*, *i.e.*, computing the point kP

where P is a point on the curve and k is an integer that is usually the secret. This operation is basically performed using the *binary algorithms* [9], which are also called *double-and-add* algorithms when used with additive groups. This operation can be performed more efficiently on Koblitz curves than on other curves.

Koblitz curves [10] are elliptic curves defined over $\mathbb{F}_2$. Their advantageous characteristic is the Frobenius mapping which can be exploited to replace the point doubling operation with a simple squaring of the point coordinates [16]. Hence, the point multiplication algorithm can be executed in a much shorter time. This technique is generally not as efficient when using an arbitrary endomorphism. In order to use this mapping efficiently, Solinas [16] has shown how to represent the scalar k in a number system of base τ, where τ is a complex number representing the squaring map. His representation is characterized by being a *non-adjacent form* where no two adjacent symbols are non-zero, in order to minimize the number of point additions. A brief background on this representation is presented in Section 2. In Section 3, we present our experimental results on an open problem proposed by Solinas. This problem questions the uniform distribution of points resulting from multiplying a randomly chosen τ-adic NAF by an input point.

In Section 4, we present an efficient algorithm that takes as input the τ-adic NAF (τNAF) representation and produces a random τ-adic representation for the same scalar value. The symbols of the randomized τ-adic representation are output one at a time from right to left which allows the execution of the right-to-left scalar multiplication along with the randomization algorithm without the need to store the new representation. The model of our algorithm has enabled us to derive a number of interesting results with regard to τ-adic representations that we present subsequently. The characteristics of τNAFs that have the maximum number of representations and formulas describing that number are presented in Section 5. The average Hamming density of the representations is derived in Section 6. Deterministic methods for determining both the average and the exact number of representations of τNAFs of a certain length are presented in Section 7. Finally, Section 8 contains the conclusion and future work.

2 Koblitz Curves and the τ-adic Representation

Koblitz curves [10]—originally named *anomalous binary curves*—are the curves E_a, $a \in \{0, 1\}$, defined over $\mathbb{F}_2$

$$E_a : y^2 + xy = x^3 + ax^2 + 1 \tag{1}$$

$E_a(\mathbb{F}_{2^m})$ is the group of $\mathbb{F}_{2^m}$-rational points on E_a. Let $\mu = (-1)^{1-a}$, that is $\mu \in \{-1, 1\}$.

The order of the group is computed as

$$\#E_a(\mathbb{F}_{2^m}) = 2^m + 1 - V_m, \tag{2}$$

where $\{V_h\}$ is the Lucas sequence defined by

$$V_0 = 2, \quad V_1 = \mu \quad \text{and} \quad V_{h+1} = \mu V_h - 2V_{h-1} \quad \text{for } h \geq 1.$$

The value of m is chosen to be a prime number so that $\#E_a(\mathbb{F}_{2^m}) = f \cdot r$ is very nearly prime, that is $r > 2$ is prime and $f = 3 - \mu$.

The main advantage of Koblitz curves when used in public-key cryptography is that scalar multiplication of the points in the main subgroup, the group of order r, can be performed without the use of point doubling operations. This is due to the following property. Since these curves are defined over $\mathbb{F}_{2^m}$, then if $P = (x, y)$ is a point on E_a, then the point (x^2, y^2) is on the curve, as well. That is the Frobenius (squaring, in this case) endomorphism $\tau : E_a(\mathbb{F}_{2^m}) \longrightarrow E_a(\mathbb{F}_{2^m})$ defined by

$$(x, y) \mapsto (x^2, y^2), \quad \mathcal{O} \mapsto \mathcal{O}$$

is well defined. It can also be verified by point addition on E_a that

$$(x^4, y^4) + 2(x, y) = \mu \cdot (x^2, y^2).$$

Hence, the squaring map can be considered as a multiplication by the complex number τ satisfying

$$\tau^2 + 2 = \mu\tau, \tag{3}$$

that is

$$\tau = \frac{1}{2}\,(\mu + \sqrt{-7}).$$

The norm of τ is 2. Thus, it is beneficial to represent the key k as an element of the ring $\mathbb{Z}[\tau]$, *i.e.*,

$$k = \sum_{i=0}^{l-1} \kappa_i \tau^i \tag{4}$$

for some l where $\deg(k) \leq l - 1$ and this representation of k is said to be of length l. We can therefore carry the scalar multiplication kP of a point P on E_a more efficiently by replacing the doubling operation in the double-and-add algorithm by the squaring map.

In [16], Solinas has shown how to represent k as in (4) in its *τ-adic non adjacent form* (τNAF) where $\kappa_i \in \{-1, 0, 1\}$ and $\kappa_i \kappa_{i+1} = 0$ for $i \geq 0$—abusing the notation, we will refer to κ_i as a signed bit or *sbit*. However, this results in $l \approx 2m$. Therefore, he proposed a *reduced*

τ*-adic non adjacent form* (RTNAF) for k where k is reduced modulo $\delta = (\tau^m - 1)/(\tau - 1)$, hence $l = m + a$. He has proven that in a τNAF representation the number of 0s is $\frac{2}{3}$ on average. He also mentioned that 1 and -1 are equally likely on average.

3 τNAFs of Length $m + a$ and their Distribution

To obtain a key represented in a reduced τNAF, we can choose an integer $k \in [1, r-1]$, and apply Solinas' method to produce its RTNAF. Alternatively, as Solinas suggests [16], we can directly choose a τNAF of length $m + a$ as follows: the first sbit is generated according to the following probability distribution

$$\kappa_i = \begin{cases} 0 & Pr(0) = 1/2 \\ 1 & Pr(1) = 1/4 \\ \bar{1} & Pr(\bar{1}) = 1/4. \end{cases} \tag{5}$$

We follow each 1 or $\bar{1}$ with a 0, and after each 0 the subsequent sbit is generated according to the distribution in (5).

This method can be verified as follows. We can consider the sequence of sbits in a random τNAF as a Markov chain of three states, namely 0, 1 and $\bar{1}$. We have the limiting probabilities as follows [16]

$$\pi_0 = 2/3 \quad \text{and} \quad \pi_1 = \pi_{\bar{1}} = 1/6. \tag{6}$$

Also, from the properties of the NAF representation, we know that a 1 or a $\bar{1}$ must be followed by a 0. Hence we have the following transition probabilities

$$P_{10} = P_{\bar{1}0} = 1 \quad \text{and} \quad P_{11} = P_{1\bar{1}} = P_{\bar{1}1} = P_{\bar{1}\bar{1}} = 0. \tag{7}$$

It remains to determine P_{00}, P_{01} and $P_{0\bar{1}}$, which we can calculate by solving the equation

$$\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi}, \tag{8}$$

where $\boldsymbol{\pi} = (\pi_0\ \pi_1\ \pi_{\bar{1}})$ and $\mathbf{P}$ is the transition matrix

$$\mathbf{P} = \begin{pmatrix} P_{00} & P_{01} & P_{0\bar{1}} \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \tag{9}$$

We obtain a unique solution to (8) which is

$$P_{00} = 1/2 \quad \text{and} \quad P_{01} = P_{0\bar{1}} = 1/4. \tag{10}$$

The sequence obtained by this method is selected from the set of all τNAFs of length $m + a$. As stated by Solinas [16], their number is the integer closest to $2^{m+a+2}/3$, whereas the order of the main subgroup is $r \approx 2^{m-2+a}$. That is the average number of sequences that, when multiplied by a given point P, would lead to the same point in the main subgroup is 16/3. The deviation from this average is an open problem. We have calculated this deviation experimentally for E_1 over small fields as follows.

We have generated all τNAFs of length $m + a$ for small m. We have then reduced each of them modulo δ, and stored how many times each of the r lattice point $\lambda_0 + \lambda_1\tau$ ($\lambda_i \in \mathbb{Z}$) in $\mathcal{V}$, which is the region spanned by the elements of $\mathbb{Z}[\tau]/\delta\mathbb{Z}[\tau]$, is mapped. The mean and standard deviation of the distribution of the number of mappings for $E_1(\mathbb{F}_{2^m})$ for small m are shown in Table 1.

Table 1: The mean and standard deviation of the number of times the lattice points of the region $\mathcal{V}$ were mapped by all τNAFs of length $m + 1$.

m	7	11	17	19	23
r	71	991	65587	262543	4196903
mean	4.803	5.511	5.329	5.325	5.330
standard deviation	0.721	0.734	0.523	0.502	0.482

As we can see from Table 1, the mean approaches $\frac{16}{3}$ as m increases. Moreover, the deviation is small and is decreasing starting from $m = 11$. Also, in our experiments the number of times a lattice point was mapped was at most 8.

4 Randomizing the τ-adic Representation of an Integer

Now, having the key k represented as a τNAF, we will present a randomization algorithm to obtain a different τ-adic representation of the key. The technique used in this algorithm is similar to the one used by Ha and Moon [6] to randomize the binary representation of the key. The difference is in the state representation which is similar to the one used in [2]. We can summarize the Ha-Moon algorithm in the following idea. A carry bit is initialized to 0 and the input binary representation is scanned, one bit at a time, starting from the least significant one. Whenever the sum of the current scanned bit and the carry is $0 \pmod 2$, the output sbit is 0, otherwise, if the sum is 1, a random decision is drawn as to whether send a 1 or a -1 to the output. In all cases the carry bit is updated properly. For example, if the current sum is 1 and the output is chosen to be -1, then 2 should be added to the remaining input bits, this is ensured by setting the carry bit to 1.

Similarly, the underlying idea of our algorithm is as follows. The sbits of the input τNAF are scanned starting from the least significant end. Whenever the scanned bit value, added to the current carry sbit, is 1, a random decision is drawn based on which the current output sbit is determined. If the latter was chosen to be 1, no change occurs in the carry sbits and the following sbit of the input as well as that of the carry are scanned. On the other hand, if the output sbit was chosen to be $\overline{1}$, this is equivalent to subtracting 2 from the current τNAF and should be compensated by adding 2 back to it. For the curve E_0, $2 = -\tau^2 - \tau = (\overline{1}\overline{1}0)_\tau$ and for the curve E_1, $2 = -\tau^2 + \tau = (\overline{1}10)_\tau$. Hence, the addition of 2 to the remaining sbits of the τNAF is handled by adding the τ-adic representation of 2 to the carry sbits. This idea is captured in the following pseudocode where the subscript τ is omitted since it is implied. The length of the output representation, the prepending of three 0s to the input as well as the number of carry sbits needed will be explained in the subsequent discussion of the algorithm implementation.

Algorithm 1. Randomization of the τ-adic representation

INPUT: $k = (\kappa_{l-1}, \dots, \kappa_1, \kappa_0)$ where k is a τNAF.

OUTPUT: $k' = (d_{l+1}, \dots, d_0)$, a random τ-adic representation of k.

1. Prepend $(\kappa_{l+2}, \kappa_{l+1}, \kappa_l) = (0, 0, 0)$ to k.

2. $(c_{2_i}, c_{1_i}, c_{0_i}) \leftarrow (0, 0, 0)$. // carry sbits.

3. for i from 0 to $l + 2$ do

   3.1 $b_i \leftarrow \kappa_i + c_{0_i}$; $r_i \leftarrow_R \{0, 1\}$. // r_i is random bit

   3.2 if $(b_i = 0)$ then
   $d_i \leftarrow 0$; $(c_{2_{i+1}}, c_{1_{i+1}}, c_{0_{i+1}}) \leftarrow (0, c_{2_i}, c_{1_i})$.

   3.3 else if $(b_i = \pm 2)$ then
   $d_i \leftarrow 0$; $(c_{2_{i+1}}, c_{1_{i+1}}, c_{0_{i+1}}) \leftarrow (0, c_{2_i}, c_{1_i}) \pm 2/\tau$.
   // for E_0, $2/\tau = (\overline{1}\overline{1})$ and for E_1, $2/\tau = (\overline{1}1)$.

   3.4 else // $b_i = \pm 1$

   3.4.1 if $(r_i = 0)$ then
   $d_i \leftarrow b_i$; $(c_{2_{i+1}}, c_{1_{i+1}}, c_{0_{i+1}}) \leftarrow (0, c_{2_i}, c_{1_i})$.

   3.4.2 else
   $d_i \leftarrow -b_i$; $(c_{2_{i+1}}, c_{1_{i+1}}, c_{0_{i+1}}) \leftarrow (0, c_{2_i}, c_{1_i}) + b_i * 2/\tau$.

As an illustration of the algorithm outcome, let $k = (10\bar{1})_\tau$ be the input τNAF, then the algorithm would output one of the following representations for the curve E_0: $(10\bar{1})_\tau, (\bar{1}\bar{1}10\bar{1})_\tau, (\bar{1}\bar{1}011)_\tau, (\bar{1}1\bar{1}1)_\tau$. Note that the τNAF is a possible output if the input sbits are sent to the output unchanged and the carry remains 0. Moreover, since the τNAF of an element of the ring $\mathbb{Z}[\tau]$ is unique [16, Theorem 1], the other representations have adjacent non-zero sbits, *i.e.*, are not τNAFs.

The algorithm can be implemented as a look-up table as in Table 2 for the curve E_1. It is also described as a *nondeterministic finite automaton* (NFA) as in Figure 1, serving the analysis in Section 7.2. As mentioned above, the sbit sequence of the key is scanned from the least significant end to the most significant end. The current state s_i is the combination of the current sbit κ_i and the carry sbits $(c_{2_i}\ c_{1_i}\ c_{0_i})_\tau$. Based on the next sbit κ_{i+1} and the random decision bit r_i, the output sbit d_i and the next state s_{i+1} are determined. Depending on whether κ_0 is $\bar{1}$, 0 or 1 the first state S_0 will be s_4, s_{12} or s_{20} respectively where the carry sbits are initialized to 0. Note that only the states in Table 2 are reachable, that is, not all combinations of the carry sbits occur in the algorithm. Moreover, by verifying the different states of the algorithm, we can observe that only three carry sbits are needed.

We will illustrate the calculation of the carry sbits and the state transitions using the following example. Let $k = (10010\bar{1})_\tau$. Then, $\kappa_0 = \bar{1}$ and $c_{2_0} = c_{1_0} = c_{0_0} = 0$ ($S_0 = s_4$). If $r_0 = 0$, $d_0 = \kappa_0 = \bar{1}$, the carry sbits do not change and the next state $S_1 = s_{12}$. Otherwise, $d_0 = 1$; to change the value of κ_0 from $\bar{1}$ to 1, we should add (-2) to the remaining sbits of k. For the curve E_1, $-2 = \tau^2 - \tau = (1\bar{1}0)_\tau$. This results in the carry sbits being $c_{2_1} = 0, c_{1_1} = 1, c_{0_1} = \bar{1}$, and the next state $S_1 = s_{14}$.

The output sbit d_i is determined by $\kappa_i + c_{0_i}$. If the latter is 0 or ± 2, then $d_i = 0$, and the carry sbits are adjusted accordingly, *e.g.*, as in the states s_2 and s_3 in Table 2. Otherwise, if $\kappa_i + c_{0_i} = \pm 1$, then if $r_i = 0$, then $d_i = \kappa_i + c_{0_i}$, else $d_i = -(\kappa_i + c_{0_i})$ and a $\pm(\bar{1}1)_\tau$ is added to $(c_{2_i}, c_{1_i})_\tau$. Note that the output d_i is determined along with the next state S_{i+1}. In other words, when the algorithm is in state S_i, the last sbit that was sent to the output is d_{i-1}.

In Figure 1, the arrows are labeled with κ_{i+1}/d_i. Solid arrows correspond to transitions where r_i is $\times$, *i.e.*, only one transition per value of κ_{i+1} is possible. Dashed arrows correspond to $r_i = 0$ and dotted arrows correspond to $r_i = 1$.

Table 2: State transition table for the randomized τ-adic representation for the curve E_1.

State					Input		Output				Next state
S_i	κ_i	c_{2_i}	c_{1_i}	c_{0_i}	κ_{i+1}	r_i	d_i	$c_{2_{i+1}}$	$c_{1_{i+1}}$	$c_{0_{i+1}}$	S_{i+1}
s_1	$\bar{1}$	0	$\bar{1}$	0	0	0	$\bar{1}$	0	0	$\bar{1}$	s_{11}
					0	1	1	1	0	0	s_{16}
s_2	$\bar{1}$	0	$\bar{1}$	1	0	$\times$	0	0	0	$\bar{1}$	s_{11}
s_3	$\bar{1}$	0	0	$\bar{1}$	0	$\times$	0	0	1	$\bar{1}$	s_{14}
s_4	$\bar{1}$	0	0	0	0	0	$\bar{1}$	0	0	0	s_{12}
					0	1	1	0	1	$\bar{1}$	s_{14}
s_5	$\bar{1}$	0	0	1	0	$\times$	0	0	0	0	s_{12}
s_6	$\bar{1}$	0	1	$\bar{1}$	0	$\times$	0	0	1	0	s_{15}
s_7	$\bar{1}$	0	1	0	0	0	$\bar{1}$	0	0	1	s_{13}
					0	1	1	0	1	0	s_{15}
s_8	0	$\bar{1}$	0	0	$\bar{1}$	$\times$	0	0	$\bar{1}$	0	s_1
					0	$\times$	0	0	$\bar{1}$	0	s_9
					1	$\times$	0	0	$\bar{1}$	0	s_{17}
s_9	0	0	$\bar{1}$	0	$\bar{1}$	$\times$	0	0	0	$\bar{1}$	s_3
					0	$\times$	0	0	0	$\bar{1}$	s_{11}
					1	$\times$	0	0	0	$\bar{1}$	s_{19}
s_{10}	0	0	$\bar{1}$	1	$\bar{1}$	0	1	0	0	$\bar{1}$	s_3
					$\bar{1}$	1	$\bar{1}$	0	$\bar{1}$	0	s_1
					0	0	1	0	0	$\bar{1}$	s_{11}
					0	1	$\bar{1}$	0	$\bar{1}$	0	s_9
					1	0	1	0	0	$\bar{1}$	s_{19}
					1	1	$\bar{1}$	0	$\bar{1}$	0	s_{17}
s_{11}	0	0	0	$\bar{1}$	$\bar{1}$	0	$\bar{1}$	0	0	0	s_4
					$\bar{1}$	1	1	0	1	$\bar{1}$	s_6
					0	0	$\bar{1}$	0	0	0	s_{12}
					0	1	1	0	1	$\bar{1}$	s_{14}
					1	0	$\bar{1}$	0	0	0	s_{20}
					1	1	1	0	1	$\bar{1}$	s_{22}
s_{12}	0	0	0	0	$\bar{1}$	$\times$	0	0	0	0	s_4
					0	$\times$	0	0	0	0	s_{12}

S_i	κ_i	c_{2_i}	c_{1_i}	c_{0_i}	κ_{i+1}	r_i	d_i	$c_{2_{i+1}}$	$c_{1_{i+1}}$	$c_{0_{i+1}}$	S_{i+1}
					1	$\times$	0	0	0	0	s_{20}
s_{13}	0	0	0	1	$\bar{1}$	0	1	0	0	0	s_4
					$\bar{1}$	1	$\bar{1}$	0	$\bar{1}$	1	s_2
					0	0	1	0	0	0	s_{12}
					0	1	$\bar{1}$	0	$\bar{1}$	1	s_{10}
					1	0	1	0	0	0	s_{20}
					1	1	$\bar{1}$	0	$\bar{1}$	1	s_{18}
s_{14}	0	0	1	$\bar{1}$	$\bar{1}$	0	$\bar{1}$	0	0	1	s_5
					$\bar{1}$	1	1	0	1	0	s_7
					0	0	$\bar{1}$	0	0	1	s_{13}
					0	1	1	0	1	0	s_{15}
					1	0	$\bar{1}$	0	0	1	s_{21}
					1	1	1	0	1	0	s_{23}
s_{15}	0	0	1	0	$\bar{1}$	$\times$	0	0	0	1	s_5
					0	$\times$	0	0	0	1	s_{13}
					1	$\times$	0	0	0	1	s_{21}
s_{16}	0	1	0	0	$\bar{1}$	$\times$	0	0	1	0	s_7
					0	$\times$	0	0	1	0	s_{15}
					1	$\times$	0	0	1	0	s_{23}
s_{17}	1	0	$\bar{1}$	0	0	0	1	0	0	$\bar{1}$	s_{11}
					0	1	$\bar{1}$	0	$\bar{1}$	0	s_9
s_{18}	1	0	$\bar{1}$	1	0	$\times$	0	0	$\bar{1}$	0	s_9
s_{19}	1	0	0	$\bar{1}$	0	$\times$	0	0	0	0	s_{12}
s_{20}	1	0	0	0	0	0	1	0	0	0	s_{12}
					0	1	$\bar{1}$	0	$\bar{1}$	1	s_{10}
s_{21}	1	0	0	1	0	$\times$	0	0	$\bar{1}$	1	s_{10}
s_{22}	1	0	1	$\bar{1}$	0	$\times$	0	0	0	1	s_{13}
s_{23}	1	0	1	0	0	0	1	0	0	1	s_{13}
					0	1	$\bar{1}$	$\bar{1}$	0	0	s_8

The algorithm keeps scanning the l sbits of the input τ-adic NAF, starting from the least significant end, moving from a state to another according to the look-up table. When the most significant sbit κ_{l-1} is reached, the algorithm is in state S_{l-1}, with the last output bit d_{l-2}.

To exit the algorithm from the state S_{l-1}, the value of the current input sbit κ_{l-1} should be added to the carry $(c_{2_{l-1}}\, c_{1_{l-1}}\, c_{0_{l-1}})_\tau$ and sent to the output. We can see from Table 2 that, for all states, the result of this addition cannot exceed three sbits. Hence, the output τ-adic representation can be of length at most $l+2$. This exit step is equivalent to prepending at most three 0s to the τNAF and continuing the algorithm as before with all subsequent random decisions $r_i = 0$. The algorithm then stops when the state s_{12} is reached, since in this state $\kappa_i = c_{2_i} = c_{1_i} = c_{0_i} = 0$. As with adding the carry to the current sbit, it can be verified from Table 2 that the paths from all states to s_{12} are at most three transitions long. We will refer to those paths as exit paths. However, from some states, there exist two exit paths that satisfy this length restriction. For example, if $S_{l-1} = s_4$, then $S_l = s_{12}$ and $d_{l-1} = \overline{1}$. Alternatively, $S_l = s_{14}$, $S_{l+1} = s_{13}$, and $S_{l+2} = s_{12}$, with the respective output $d_{l-1} = 1$, $d_l = \overline{1}$, $d_{l+1} = 1$. Other states that have two possible exit paths are $s_7, s_{10}, s_{11}, s_{13}, s_{14}, s_{17}$ and s_{20}.

The same randomization technique can be applied to the τ-adic representation of integers when the points are on the curve E_0. In this case, $2 = -\tau^2 - \tau = (\overline{1}\overline{1}0)_\tau$, which will produce different carry sbits than for the curve E_1, and hence different states. Those states and the transitions between them are listed in Table 3. For this curve, the states that have two possible exit paths are $s_2, s_4, s_9, s_{11}, s_{13}, s_{15}, s_{20}$ and s_{22}. We have included the representations of the τNAFs of length $1 \le l \le 4$ on the curve E_0 in Appendix A. As can be seen in this appendix, the number of representations is not uniform among the τNAFs, which is expected to be true for any length l. Hence, it is not favorable to choose a key by choosing a random τ-adic expansion.

Table 3: State transition table for the randomized τ-adic representation for the curve E_0.

State					Input		Output				Next state
S_i	κ_i	c_{2_i}	c_{1_i}	c_{0_i}	κ_{i+1}	r_i	d_i	$c_{2_{i+1}}$	$c_{1_{i+1}}$	$c_{0_{i+1}}$	S_{i+1}
s_1	$\overline{1}$	0	$\overline{1}$	$\overline{1}$	0	$\times$	0	0	1	0	s_{14}
s_2	$\overline{1}$	0	$\overline{1}$	0	0	0	$\overline{1}$	0	0	$\overline{1}$	s_{11}
					0	1	1	0	1	0	s_{14}
s_3	$\overline{1}$	0	0	$\overline{1}$	0	$\times$	0	0	1	1	s_{15}
s_4	$\overline{1}$	0	0	0	0	0	$\overline{1}$	0	0	0	s_{12}
					0	1	1	0	1	1	s_{15}
s_5	$\overline{1}$	0	0	1	0	$\times$	0	0	0	0	s_{12}
s_6	$\overline{1}$	0	1	0	0	0	$\overline{1}$	0	0	1	s_{13}
					0	1	1	$\overline{1}$	0	0	s_8

S_i	κ_i	c_{2_i}	c_{1_i}	c_{0_i}	κ_{i+1}	r_i	d_i	$c_{2_{i+1}}$	$c_{1_{i+1}}$	$c_{0_{i+1}}$	S_{i+1}
s_7	$\bar{1}$	0	1	1	0	$\times$	0	0	0	1	s_{13}
s_8	0	$\bar{1}$	0	0	$\bar{1}$	$\times$	0	0	$\bar{1}$	0	s_2
					0	$\times$	0	0	$\bar{1}$	0	s_{10}
					1	$\times$	0	0	$\bar{1}$	0	s_{18}
s_9	0	0	$\bar{1}$	$\bar{1}$	$\bar{1}$	0	$\bar{1}$	0	0	$\bar{1}$	s_3
					$\bar{1}$	1	1	0	1	0	s_6
					0	0	$\bar{1}$	0	0	$\bar{1}$	s_{11}
					0	1	1	0	1	0	s_{14}
					1	0	$\bar{1}$	0	0	$\bar{1}$	s_{19}
					1	1	1	0	1	0	s_{22}
s_{10}	0	0	$\bar{1}$	0	$\bar{1}$	$\times$	0	0	0	$\bar{1}$	s_3
					0	$\times$	0	0	0	$\bar{1}$	s_{11}
					1	$\times$	0	0	0	$\bar{1}$	s_{19}
s_{11}	0	0	0	$\bar{1}$	$\bar{1}$	0	$\bar{1}$	0	0	0	s_4
					$\bar{1}$	1	1	0	1	1	s_7
					0	0	$\bar{1}$	0	0	0	s_{12}
					0	1	1	0	1	1	s_{15}
					1	0	$\bar{1}$	0	0	0	s_{20}
					1	1	1	0	1	1	s_{23}
s_{12}	0	0	0	0	$\bar{1}$	$\times$	0	0	0	0	s_4
					0	$\times$	0	0	0	0	s_{12}
					1	$\times$	0	0	0	0	s_{20}
s_{13}	0	0	0	1	$\bar{1}$	0	1	0	0	0	s_4
					$\bar{1}$	1	$\bar{1}$	0	$\bar{1}$	$\bar{1}$	s_1
					0	0	1	0	0	0	s_{12}
					0	1	$\bar{1}$	0	$\bar{1}$	$\bar{1}$	s_9
					1	0	1	0	0	0	s_{20}
					1	1	$\bar{1}$	0	$\bar{1}$	$\bar{1}$	s_{17}
s_{14}	0	0	1	0	$\bar{1}$	$\times$	0	0	0	1	s_5
					0	$\times$	0	0	0	1	s_{13}
					1	$\times$	0	0	0	1	s_{21}
s_{15}	0	0	1	1	$\bar{1}$	0	1	0	0	1	s_5
					$\bar{1}$	1	$\bar{1}$	0	$\bar{1}$	0	s_2
					0	0	1	0	0	1	s_{13}
					0	1	$\bar{1}$	0	$\bar{1}$	0	s_{10}
					1	0	1	0	0	1	s_{21}
					1	1	$\bar{1}$	0	$\bar{1}$	0	s_{18}

S_i	κ_i	c_{2_i}	c_{1_i}	c_{0_i}	κ_{i+1}	r_i	d_i	$c_{2_{i+1}}$	$c_{1_{i+1}}$	$c_{0_{i+1}}$	S_{i+1}
s_{16}	0	1	0	0	$\bar{1}$	$\times$	0	0	1	0	s_6
					0	$\times$	0	0	1	0	s_{14}
					1	$\times$	0	0	1	0	s_{22}
s_{17}	1	0	$\bar{1}$	$\bar{1}$	0	$\times$	0	0	$\bar{1}$	$\bar{1}$	s_{11}
s_{18}	1	0	$\bar{1}$	0	0	0	1	0	0	$\bar{1}$	s_{11}
					0	1	$\bar{1}$	1	0	0	s_{16}
s_{19}	1	0	0	$\bar{1}$	0	$\times$	0	0	0	0	s_{12}
s_{20}	1	0	0	0	0	0	1	0	0	0	s_{12}
					0	1	$\bar{1}$	0	$\bar{1}$	$\bar{1}$	s_9
s_{21}	1	0	0	1	0	$\times$	0	0	$\bar{1}$	$\bar{1}$	s_9
s_{22}	1	0	1	0	0	0	1	0	0	1	s_{13}
					0	1	$\bar{1}$	0	$\bar{1}$	0	s_{10}
s_{23}	1	0	1	1	0	$\times$	0	0	$\bar{1}$	0	s_{10}

The Ha-Moon randomization algorithm [6] was proposed as a countermeasure to differential power analysis (DPA) attacks on ECCs. The output of the algorithm was a random binary signed-digit (BSD) representation of the input binary representation of the key. The resulting BSD representation is then used by the binary algorithm [9, Section 4.6.3] allowing negative digits. Later on, it was shown in [4, 13], that this randomization method does not serve its purpose since the number of intermediate points possibly computed at any iteration of the binary algorithm is only two. Guessing the value of an intermediate point is at the core of a conventional DPA attack. The reason behind this limited number of intermediate points is that the following relation always hold for some key $k = (k_{n-1}, \ldots, k_0)_2$ with BSD representation $k' = (k'_n, \ldots, k'_0)_2$, where $k'_i \in \{-1, 0, 1\}$

$$\sum_{i=0}^{j-1} k_i 2^i = \sum_{i=0}^{j-1} k'_i 2^i + c_j 2^j, \tag{11}$$

for any $0 < j \leq n$, where $c_j \in \{0, 1\}$ is the carry bit in the Ha-Moon algorithm. The carry takes only one of two values, and so does the intermediate point computed by the binary algorithm using the BSD representation of the key.

Similar arguments apply to the τ-adic representation. However, from Tables 2 and 3, we can see that the carry sbits can take one of 9 possible values. Hence, the adequacy of this randomization method as a DPA countermeasure depends on the application and the life length of the key. It is interesting to study the probability of occurrence of a certain intermediate value of the representation in relation to the sbits values of the original τ-NAF similar to the study presented by [4] on the BSD representation. It is also interesting to investigate the number of carry patterns that would result if the input is the shortest τ-adic representation of an element of $\mathbb{Z}[\tau]$ [17]. Note that in this case, the number of carry sbits may be more than 3 and it is not guaranteed that the number of states is finite.

5 τNAF with the Maximum Number of Representations

Let k be a τNAF of length l sbits, possibly having 0(s) as the leading sbit(s), and let $\vartheta(k,l)$ be the number of τ-adic representations of k. Note that those representations are of length at most $l+2$ as in Section 4. In the following, we will focus our discussion on "positive" τNAFs, *i.e.*, those having $\kappa_{l-1} = \kappa_{l-2} = \ldots = \kappa_i = 0$ and $\kappa_{i-1} = 1$ for some $0 < i \leq l$. Since $-k$ is obtained from k by interchanging the $\bar{1}$s with the 1s, in the same way, the representations of $-k$ can be obtained from those of k, hence, $\vartheta(k,l) = \vartheta(-k,l)$. Let $k_{max,l}$ be the τNAF of length l that has the maximum number of representations among other τNAFs of the same length (cf. Table 8 in Appendix A). Also, let $\alpha(k,l')$ be the number of representations of a τNAF k that are of length at most l' sbits. According to these definitions, for any τNAF k of length up to l, we have

$$\vartheta(k,l) = \alpha(k,l+2) \leq \vartheta(k_{max,l},l).$$

For example, from Table 7, we have $\alpha((101)_\tau,2) = 1, \alpha((101)_\tau,3) = 2, \ldots, \alpha((101)_\tau,6) = 7 < \vartheta((10\bar{1})_\tau,4) = 8$.

Now we state the following theorem.

Theorem 1 *Let $l \geq 1$ and $w = \lfloor \frac{l-1}{2} \rfloor$. For l odd,*

$$k_{max,l} = \tau^{2w} + \sum_{i=0}^{w-1} (-1)^{w-1-i} \tau^{2i}. \tag{12}$$

For l even,

$$k_{max,l} = \sum_{i=0}^{w} (-1)^{w-i} \tau^{2i}. \tag{13}$$

And for any τNAF k of length up to $l+3$,

$$\alpha(k,l+2) \leq \vartheta(k_{max,l},l). \tag{14}$$

Moreover, for $l \geq 3$,

$$\vartheta(k_{max,l},l) = \vartheta(k_{max,l-1},l-1) + \vartheta(k_{max,l-2},l-2). \tag{15}$$

In order to prove the theorem, we will use the following lemmas.

Lemma 1 *If k is divisible by τ^e then $\vartheta(k,l) = \vartheta\left(\frac{k}{\tau^e}, l-e\right)$.*

Proof. Looking at Table 2 and Table 3, we find that random decisions are made at the states where $\kappa_i + c_{0_i} = \pm 1$. In this case, there are two possible transitions emerging from these states, that is there are two possible paths that can be followed, each yielding a family of representations where the sbit d_i is either 1 or $\bar{1}$.

When the least significant sbit(s) is (are) 0, the algorithm enters state s_{12} and does not exit this state until the first 1 or $\bar{1}$ is encountered. Until then, there are no new representations that are formed, and the least significant 0s are sent to the output as they are. Any other representation formed thereafter will have the same number of least significant 0s as k.

In other words, if k is divisible by τ^e, so are its representations. That is, they will all have e least significant 0s. Therefore, the possible representations for k when represented in l sbits will be the same representations for $\frac{k}{\tau^e}$ when represented in $l-e$ sbits with e 0s appended to each of the latters. $\square$

Lemma 2 *If k is a τNAF of length l and $k \equiv (-1)^b \pmod{\tau}$ where $b \in \{0,1\}$, then the τNAF of $k+(-1)^b$ is of length at most $l+3$.*

Proof. To convert a number in a τ-adic form into a τNAF, we can use the transformations given by Gordon [5] for the curve E_1. The following transformations (and their negatives) are the equivalent ones for the curve E_0.

$$\tau + 1 \rightarrow -\tau^2 - 1 \quad (11 \rightarrow \bar{1}0\bar{1}), \tag{16}$$

$$\tau - 1 \rightarrow -\tau^3 + 1 \quad (1\bar{1} \rightarrow \bar{1}001), \tag{17}$$

$$2 \rightarrow \tau^3 + \tau \quad (2 \rightarrow 1010). \tag{18}$$

Now, consider the following cases for the least significant sbits of $k \equiv 1 \pmod{\tau}$ when 1 is added, where the transformation (18) is used after the addition. Other cases are recursions of the following ones. The subscript τ was removed since it applies to all of the following representations.

$(\ldots\bar{1}001) + 1 = (\ldots 0010)$,

$(\ldots 1001) + 1 = (\ldots 2010)$,

$(\ldots 0101) + 1 = (\ldots 1110) = (\ldots 00\bar{1}0)$, using (16)

$(\ldots\bar{1}0\bar{1}01) + 1 = (\ldots\bar{1}1\bar{1}10) = (\ldots 010\bar{1}0)$, using -(17) (*i.e.*, the negative of (17))

$(\ldots 10\bar{1}01) + 1 = (\ldots 11\bar{1}10) = (\ldots 210\bar{1}0)$, using -(17)

$(\ldots 100\bar{1}01) + 1 = (\ldots 101\bar{1}10) = (\ldots 1110\bar{1}0) = (\ldots 00\bar{1}0\bar{1}0)$, using -(17) and (16)

$(\ldots \bar{1}00\bar{1}01) + 1 = \ldots = (\bar{2}0\bar{1}0\bar{1}0)$, using -(17) and (16).

When any of the transformations (16) to (18) is used, the resulting carry will either cancel an existing sbit, be added to a 0 or result in a 2 or -2. We can see from the above cases that the absolute result of adding a carry to an sbit will not exceed 2. Thus, the resulting τNAF of $k+1$ is at most 3 sbits longer than k. The same argument applies to $k \equiv -1 \pmod{\tau}$. $\square$

Lemma 3 *For any τNAF $k \equiv (-1)^b \pmod{\tau}$ of length l, where $b \in \{0,1\}$, we have*

$$\vartheta(k,l) = \vartheta\left(\frac{k-(-1)^b}{\tau^2}, l-2\right) + \alpha\left(\frac{k+(-1)^b}{\tau}, l+1\right),$$

where $k + (-1)^b$ is in τNAF representation.

Proof. We will consider here the case of $k \equiv 1 \pmod{\tau}$ but the same arguments apply to $k \equiv -1 \pmod{\tau}$. Recall that $\vartheta(k,l)$ is the number of representations of k that are of length at most $l+2$. Since $k \mod \tau \neq 0$, this is also true for the τ-adic representations of k. That is, their least significant sbit (LSSB) will be either 1 or $\bar{1}$. For those representations that have 1 as the LSSB, if this 1 is replaced with 0, they will become representations of $k-1$. Since k is a τNAF, then $k-1$ is a τNAF divisible by τ^2. From Lemma 1, we know that the number of representations of $k-1$ is $\vartheta(k-1,l) = \vartheta\left(\frac{k-1}{\tau^2}, l-2\right)$ and that those representations will have their 2 LSSBs equal to 00. Therefore, they can all be used as representations of k by replacing the least significant 0 with 1.

On the other hand, for those representations that have $\bar{1}$ as their LSSB, if this $\bar{1}$ is replaced with 0, they will become representations of $k+1$. Since $2 = (\bar{1}10)_\tau$ for the curve E_1 and $2 = (\bar{1}\bar{1}0)_\tau$ for the curve E_0, we can see that $k+1 \equiv 0 \pmod{\tau}$, hence all the representations of $k+1$ have 0 as their LSSB. Those representations that are of length $l+2$, with their least significant 0 replaced with $\bar{1}$, are counted among the $\vartheta(k,l)$ representations of k and their number is $\alpha(k+1, l+2) = \alpha\left(\frac{k+1}{\tau}, l+1\right)$, where the equality follows from Lemma 1. $\square$

The following lemmas are carried on E_0 but there exist corresponding lemmas on E_1.

Lemma 4 *For l odd and $w = \frac{l-1}{2}$, if $k = \tau^{2w} + \sum_{i=0}^{w-1}(-1)^{w-1-i}\tau^{2i}$, then $\sum_{i=0}^{w-1}(-1)^{w-i}\tau^{2i+1} + (-1)^w$ is among the representations of k. In other words, $\frac{k-(-1)^w}{\tau} = \frac{k+(-1)^{w-1}}{\tau} = \sum_{i=0}^{w-1}(-1)^{w-i}\tau^{2i}$.*

Proof. Without loss of generality, let w be odd, then $k = (1\ 0\ 1\ 0\ \bar{1}\ 0\ \ldots\ 1\ 0\ \bar{1}\ 0\ 1)_\tau$. When the least significant 1 is replaced by $\bar{1}$, $2 = (\bar{1}\bar{1}0)_\tau$ is added to k. Hence,

$$\begin{aligned} k &= (1\,0\,1\,0\,\bar{1}\,0\,\ldots\,1\,0\,\bar{2}\,\bar{1}\,\bar{1})_\tau \\ &= (1\,0\,1\,0\,\bar{1}\,0\,\ldots\,2\,1\,0\,\bar{1}\,\bar{1})_\tau \\ &= \ldots \\ &= (1\,0\,1\,0\,\bar{2}\,\bar{1}\,\ldots\,0\,1\,0\,\bar{1}\,\bar{1})_\tau \\ &= (1\,0\,2\,1\,0\,\bar{1}\,\ldots\,0\,1\,0\,\bar{1}\,\bar{1})_\tau \\ &= (0\,\bar{1}\,0\,1\,0\,\bar{1}\,\ldots\,0\,1\,0\,\bar{1}\,\bar{1})_\tau. \end{aligned}$$

□

Lemma 5 *For l even and $w = \lfloor \frac{l-1}{2} \rfloor = \frac{l}{2} - 1$, if $k = \sum_{i=0}^{w} (-1)^{w-i}\tau^{2i}$, then $\tau^{2w+3} + \tau^{2w+1} + \sum_{i=0}^{w-1} (-1)^{w-1-i}\tau^{2i+1} + (-1)^{w-1}$ is among the representations of k. In other words, $\frac{k-(-1)^{w-1}}{\tau} = \frac{k+(-1)^{w}}{\tau} = \tau^{2w+2} + \tau^{2w} + \sum_{i=0}^{w-1}(-1)^{w-1-i}\tau^{2i}$*

Proof. Without loss of generality, let w be odd. Then, k is of the form $(0\,1\,0\,\bar{1}\,0\,\ldots\,1\,0\,\bar{1})_\tau$. As before, the least significant $\bar{1}$ can be replaced by 1 and $-2 = (110)_\tau$ added to k. Hence, we obtain the following

$$\begin{aligned} k = &\quad (0\,1\,0\,\bar{1}\,0\,\ldots\,2\,1\,1)_\tau \\ = &\quad \ldots \\ = &\quad (0\,1\,0\,\bar{2}\,\bar{1}\,\ldots\,0\,1\,1)_\tau \\ = &\quad (0\,2\,1\,0\,\bar{1}\,\ldots\,0\,1\,1)_\tau \\ = &\quad (\bar{1}\,\bar{1}\,0\,1\,0\,\bar{1}\,\ldots\,0\,1\,1)_\tau \\ = &\quad (1\,0\,1\,0\,1\,0\,\bar{1}\,\ldots\,0\,1\,1)_\tau. \end{aligned}$$

□

Lemma 6 *Let k be τNAF of length l with $\kappa_{l-1} = 1$ $(\bar{1})$. Then, the representations of k that are of length $l+2$ will have $d_{l+1} = \bar{1}$ (1), where d_i are the sbits output from the algorithm as in Table 3. Moreover, if $d_{l-1} = \bar{1}$ in any of the representations of k, then the length of this representation is $l+2$.*

Proof. Considering Table 3, when the most significant sbit $\kappa_{l-1} = 1$ is read, the algorithm will be in one of the states s_{17} to s_{23}. Representations that are of length $l+2$ are resulting from those states that have exit paths consisting of three transitions as single exit paths (s_{21} and s_{23}) or as alternate paths (s_{20} and s_{22}). It can be easily checked from the table that the last output sbit in all such paths is $\bar{1}$. It can be also checked that $d_{l-1} = \bar{1}$ occurs only on the alternate exit paths from s_{20} and s_{22}, hence the second part of the lemma is proved. The same arguments applies for $\kappa_{l-1} = \bar{1}$. □

Now we employ the previous lemmas to prove Theorem 1 by induction.

Proof of Theorem 1. From the algorithm using Table 3, we can verify the following (cf. Tables 4 to 7 in Appendix A):

- $\vartheta((1)_\tau, 1) = 2$, those two representations are $(1)_\tau, (\overline{1}1\overline{1})_\tau$. $k_{max,1} = 1$.
- $\vartheta((1)_\tau, 2) = 3$, those representations are $(1)_\tau, (\overline{1}1\overline{1})_\tau, (101\overline{1})_\tau$. From Lemma 1, we have $\vartheta((10)_\tau, 2) = \vartheta((1)_\tau, 1) = 2$. So, $k_{max,2} = 1$.
- $\vartheta((101)_\tau, 3) = 5$. $k_{max,3} = 101$. The five representations are $(101)_\tau$, $(\overline{1}1\overline{1}01)\tau$, $(\overline{1}\overline{1})_\tau$, $(111\overline{1})_\tau$, $(\overline{1}0\overline{1}1\overline{1})_\tau$. The first two representations are the same representations of $(100)_\tau$ for $l = 3$, with 1 as the least significant sbit instead of 0. From Lemma 1, we have $\vartheta((100)_\tau, 3) = \vartheta((1)_\tau, 1) = 2$. The remaining three representations are the same representations of $(\overline{1})_\tau$ for $l = 2$ shifted left by τ with $\overline{1}$ added. Note that the representations of $\overline{1}$ are the negative of the representations of 1. Hence, $\vartheta((101)_\tau, 3) = \vartheta((1)_\tau, 2) + \vartheta((1)_\tau, 1)$.
- For $l = 1$, we can see from Table 7 that for all τNAFs k of length up to $l + 3 = 4$, $\alpha(k, 3) \leq \vartheta(k_{max,1}, 1)$.

We see that, in Theorem 1, (12) and (14) are true for $l = 1$, (13) is true for $l = 2$ and (15) is true for $l = 3$. Now assume that the theorem is true up to some length $l - 1$.

From Lemma 1, $k_{max,l} \equiv (-1)^b \pmod{\tau}$, for $b \in \{0, 1\}$. From Lemma 3, we know that

$$\vartheta(k_{max,l}, l) = \vartheta\left(\frac{k_{max,l} - (-1)^b}{\tau^2}, l - 2\right) + \alpha\left(\frac{k_{max,l} + (-1)^b}{\tau}, l + 1\right). \tag{19}$$

From Lemma 2, we know that $\frac{k_{max,l}+(-1)^b}{\tau}$ will be of length at most $l + 2$ and, based on our assumption, for any τNAF k of length up to $l + 2$, $\alpha(k, l + 1) \leq \vartheta(k_{max,l-1}, l - 1)$ is true.

Let l be odd and k' of length l be equal to $\tau^{2w} + \sum_{i=0}^{w-1}(-1)^{w-1-i}\tau^{2i}$ where $w = \frac{l-1}{2}$, that is $k' \equiv (-1)^{w-1} \pmod{\tau}$. Then, we have $\frac{k'-(-1)^{w-1}}{\tau^2} = \tau^{2(w-1)} + \sum_{i=0}^{w-2}(-1)^{w-2-i}\tau^{2i} = k_{max,l-2}$ (the last equality follows from our assumption that (12) in Theorem 1 is true up to τNAFs of length $l - 1$). Also, from Lemma 4, we have $\frac{k'+(-1)^{w-1}}{\tau}$ is equivalent to $\sum_{i=0}^{w-1}(-1)^{w-i}\tau^{2i} = -k_{max,l-1}$. Since $\alpha(-k_{max,l-1}, l + 1) = \vartheta(-k_{max,l-1}, l - 1) = \vartheta(k_{max,l-1}, l - 1)$, then both terms of (19) are maximal and, hence, $k' = k_{max,l}$, proving (12).

Now, let l be even and k' of length l be equal to $\sum_{i=0}^{w}(-1)^{w-i}\tau^{2i}$ where $w = \lfloor\frac{l-1}{2}\rfloor = \frac{l}{2} - 1$, that is $k' \equiv (-1)^w \pmod{\tau}$. Then, we have $\frac{k'-(-1)^w}{\tau^2} = \sum_{i=0}^{w-1}(-1)^{w-1-i}\tau^{2i} = k_{max,l-2}$. Also, from Lemma 5, $\frac{k'+(-1)^w}{\tau}$ is equivalent to $\tau^{2w+2} + \tau^{2w} + \sum_{i=0}^{w-1}(-1)^{w-1-i}\tau^{2i} = \tau^{2w+2} + k_{max,l-1}$. According to Lemma 6, the representations of $k_{max,l-1}$ that are of length $l + 1$ have their most significant term equal to $-\tau^{2w+2}$. Therefore, all the representations of $\tau^{2w+2} + k_{max,l-1}$ will be of length at most $l+1$ and can be used as representations for k' by shifting them to the left by one sbit and adding to them $(-1)^{w-1}$. Hence, $\alpha(\tau^{2w+2} + k_{max,l-1}, l+1) = \vartheta(k_{max,l-1}, l-1)$, and $k' = k_{max,l}$, proving (13).

From the previous discussion and (19), we can see that (15) is true.

Now, it remains to prove (14), that is for all τNAFs k of length up to $l+3$,

$$\alpha(k, l+2) \leq \vartheta(k_{max,l}, l).$$

We have already assumed that for any τNAF k of length up to $l+2$, $\alpha(k, l+1) \leq \vartheta(k_{max,l-1}, l-1) < \vartheta(k_{max,l}, l)$ is true, where the last inequality follows from (15). Now, let k be a τNAF of length $l+3$. If $k \equiv 0 \pmod{\tau}$, from Lemma 1 we have,

$$\begin{aligned}
\alpha(k, l+2) &= \alpha\left(\frac{k}{\tau}, l+1\right) \\
&\leq \vartheta(k_{max,l-1}, l-1), \text{by assumption} \\
&< \vartheta(k_{max,l}, l).
\end{aligned}$$

Otherwise, if $k \equiv (-1)^b \pmod{\tau}$, then some of the representations of k will have 1 as their LSSB and the others will have $\bar{1}$. Without loss of generality, let $b = 0$. From Lemma 3, the representations that end with 1 and are of length $l+2$, are those of $\frac{k-1}{\tau^2}$ that are of length l with an appended 01. Hence, their number is $\alpha\left(\frac{k-1}{\tau^2}, l\right) \leq \vartheta(k_{max,l-2}, l-2)$. On the other hand, the representations of k that end with $\bar{1}$ and are of length $l+2$ are those of $\frac{k+1}{\tau}$ that are of length $l+1$ with an appended $\bar{1}$. Their number is $\alpha(\frac{k+1}{\tau}, l+1) \leq \vartheta(k_{max,l-1}, l-1)$. Note that $\frac{k-1}{\tau^2}$ and $\frac{k+1}{\tau}$ are τNAFs of length $l+1$ and $l+2$, respectively. Hence, we have

$$\begin{aligned}
\alpha(k, l+2) &= \alpha(\frac{k-1}{\tau^2}, l) + \alpha(\frac{k+1}{\tau}, l+1) \\
&\leq \vartheta(k_{max,l-2}, l-2) + \vartheta(k_{max,l-1}, l-1) \\
&\leq \vartheta(k_{max,l}, l).
\end{aligned}$$

□

$\vartheta(k_{max,l}, l)$ as a Fibonacci Number

The Fibonacci numbers form a sequence defined by the following recurrence relation [11]

$$F(0) = 0, \quad F(1) = 1, \quad F(l) = F(l-1) + F(l-2),\ l > 1. \tag{20}$$

The closed-form expression of Fibonacci numbers, which is known as Binet's formula, is

$$F(l) = \frac{\varphi^l - (1-\varphi)^l}{\sqrt{5}}, \tag{21}$$

where

$$\varphi = \frac{1+\sqrt{5}}{2}, \tag{22}$$

φ is known as *the golden ratio.*

From Theorem 1, we can see that the values of $\vartheta(k_{max,l}, l)$ for $l \geq 1$ form a Fibonacci sequence, where from Tables 4 and 5, we have

$$\vartheta(k_{max,1}, 1) = 2 = F(3),$$
$$\vartheta(k_{max,1}, 2) = 3 = F(4),$$

hence,

$$\vartheta(k_{max,l}, l) = F(l+2) = \frac{\varphi^{l+2} - (1-\varphi)^{l+2}}{\sqrt{5}}. \tag{23}$$

It is also important to notice that the recurrence relation of $\vartheta(k_{max,l}, l)$ in Theorem 1 is identical to the recurrence we obtained for the maximum number of binary signed digit (BSD) representations of an integer [3, Lemma 6]. Since the values of $\vartheta(k_{max,l}, l)$ for $l = 1, 2$ agree with the values of $\delta(k_{max,n}, n)$ for $n = 1, 2$, respectively, in the BSD system, then the formula we obtained for $\delta(k_{max,n}, n)$ is directly applicable to the τ-adic representation system. That is, for l even, let $m = \frac{l}{2}$, then we have

$$\begin{aligned}\vartheta(k_{max,l}, l) = 3^m - (m-1)3^{m-2} + \left(\sum_{i_1=1}^{m-3} i_1\right) 3^{m-4} \\ - \left(\sum_{i_1=1}^{m-5}\sum_{i_2=1}^{i_1} i_2\right) 3^{m-6} + \left(\sum_{i_1=1}^{m-7}\sum_{i_2=1}^{i_1}\sum_{i_3=1}^{i_2} i_3\right) 3^{m-8} - \cdots .\end{aligned} \tag{24}$$

And for l odd, with $m = \frac{l-1}{2}$, we have

$$\begin{aligned}\vartheta(k_{max,l}, l) = 2 \cdot 3^m - \left[3^{m-1} + 2(m-1)3^{m-2}\right] \\ + \left[(m-2)3^{m-3} + 2\left(\sum_{i_1=1}^{m-3} i_1\right) 3^{m-4}\right] \\ - \left[\left(\sum_{i_2=1}^{m-4} i_2\right) 3^{m-5} + 2\left(\sum_{i_1=1}^{m-5}\sum_{i_2=1}^{i_1} i_2\right) 3^{m-6}\right] \\ + \left[\left(\sum_{i_2=1}^{m-6}\sum_{i_3=1}^{i_2} i_3\right) 3^{m-7} + 2\left(\sum_{i_1=1}^{m-7}\sum_{i_2=1}^{i_1}\sum_{i_3=1}^{i_2} i_3\right) 3^{m-8}\right] \\ - \cdots\end{aligned} \tag{25}$$

From the latter expressions, $\vartheta(k_{max,l}, l)$ appears to be $\mathcal{O}(3^{\lfloor \frac{n}{2} \rfloor})$. However, a tighter bound is obtained from (23) and is given by $\mathcal{O}(\varphi^n)$, where the golden ratio $\varphi \approx 1.618 < 3^{\frac{1}{2}}$.

On a related note, Equations (24) and (25) can be considered another solution for the Fibonacci number $F(n-2)$ (cf. (23)). The first few terms of these formulas can then be used as an approximation to a Fibonacci number where floating point arithmetic is not available.

Corollary 1 *$k_{max,l}$ is unique among positive τNAFs of length up to l.*

Proof. We will carry the proof by induction. We can see that this is true for $l = 1$ and 2 from Tables 4 and 5, respectively. Now we assume that it is true up to some length $l-1$.

We assume that $k_{max_1,l} = k_{max,l}$ as defined by (12) and (13). We want to find a τNAF$k_{max_2,l} \neq k_{max_1,l}$ such that $\vartheta(k_{max_2,l}) = \vartheta(k_{max_1,l})$. We know that $k_{max_1,l} \equiv (-1)^b \pmod{\tau}$ where $b \in \{0, 1\}$ and that $\frac{k_{max_1,l} - (-1)^b}{\tau^2} = k_{max,l-2}$, then $k_{max_2,l} = \tau^2 k_{max,l-2} - (-1)^b = \tau^2 k_{max,l-2} + (-1)^{b-1} = k_{max_1,l} - 2(-1)^b$ where the first equality follows from Lemma 3 and assuming $k_{max,l-2}$ is unique, *i.e.*, the only τNAFwith maximum number of representations among τNAFs of length $l-2$.

Let l be odd, then $k_{max_2,l} = \tau^{2w} + \sum_{i=1}^{w-1}(-1)^{w-1-i}\tau^{2i} + (-1)^{w-2}$. According to Lemma 3 and Theorem 1, if $\alpha(\frac{k_{max_2,l}+(-1)^{w-2}}{\tau}, l+1) = \vartheta(k_{max,l-1}, l-1)$, then $\vartheta(k_{max_2,l}) = \vartheta(k_{max_1,l})$. However,

$$\begin{aligned} k_{max_2,l} + (-1)^{w-2} &= \tau^{2w} + \sum_{i=1}^{w-1}(-1)^{w-1-i}\tau^{2i} + 2(-1)^{w-2} \\ &= \tau^{2w} + \sum_{i=2}^{w-1}(-1)^{w-1-i}\tau^{2i} + (-1)^{w-2}\tau^2 + (-\tau^2 - \tau)(-1)^{w-2} \\ &= \tau^{2w} + \sum_{i=2}^{w-1}(-1)^{w-1-i}\tau^{2i} - \tau(-1)^{w-2}, \end{aligned}$$

where the second equality follows from the fact $2 = -\tau^2 - \tau$ on the curve E_0. Note that $\frac{k_{max_2,l}+(-1)^{w-2}}{\tau}$ is actually a τNAF of length $l-1$ that is not equal to $k_{max,l-1}$. The same argument applies to the curve E_1 where $2 = -\tau^2 + \tau$.

Now let l be even, then $k_{max_2,l} = \sum_{i=1}^{w}(-1)^{w-i}\tau^{2i} + (-1)^{w-1}$. We have (for the curve E_0)

$$\begin{aligned} k_{max_2,l} + (-1)^{w-1} &= \sum_{i=1}^{w}(-1)^{w-i}\tau^{2i} + 2(-1)^{w-1}. \\ &= \sum_{i=2}^{w}(-1)^{w-i}\tau^{2i} + (-1)^{w-1}\tau^2 + (-\tau^2 - \tau)(-1)^{w-1} \\ &= \sum_{i=2}^{w}(-1)^{w-i}\tau^{2i} - (-1)^{w-1}\tau, \end{aligned}$$

which is τNAF of length $l-2$. $\square$

6 Average Hamming Density of the Representations

We assume that the τNAF k has been randomly chosen among all τNAFs of length $m+a$ as was suggested by Solinas [16]. Since the decision bit r_i is also randomly chosen, the transition from a state S_i to the next state S_{i+1} does not depend on the previous states $S_{i-1}, S_{i-2}, \ldots$. Thus, this process is a finite Markov chain. Also it is irreducible, since every state is reachable from every other state in a finite number of steps. And it is ergodic, as it has recurrent aperiodic states[1]. Therefore, the limiting probabilities of all states can be calculated using (8).

We can write the transition matrix for the states of Table 2 as follows

$$\mathbf{T} = \begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&\frac{1}{2}&0&0&0&0&\frac{1}{2}&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&\frac{1}{2}&0&\frac{1}{2}&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&\frac{1}{2}&0&\frac{1}{2}&0&0&0&0&0&0&0&0\\
\frac{1}{4}&0&0&0&0&0&0&0&\frac{1}{2}&0&0&0&0&0&0&0&\frac{1}{4}&0&0&0&0&0&0\\
0&0&\frac{1}{4}&0&0&0&0&0&0&0&\frac{1}{2}&0&0&0&0&0&0&0&\frac{1}{4}&0&0&0&0\\
\frac{1}{8}&0&\frac{1}{8}&0&0&0&0&0&\frac{1}{4}&0&\frac{1}{4}&0&0&0&0&0&\frac{1}{8}&0&\frac{1}{8}&0&0&0&0\\
0&0&0&\frac{1}{8}&0&\frac{1}{8}&0&0&0&0&0&\frac{1}{4}&0&\frac{1}{4}&0&0&0&0&0&\frac{1}{8}&0&\frac{1}{8}&0\\
0&0&0&\frac{1}{4}&0&0&0&0&0&0&0&\frac{1}{2}&0&0&0&0&0&0&0&\frac{1}{4}&0&0&0\\
0&\frac{1}{8}&0&\frac{1}{8}&0&0&0&0&0&\frac{1}{4}&0&\frac{1}{4}&0&0&0&0&0&\frac{1}{8}&0&\frac{1}{8}&0&0&0\\
0&0&0&0&\frac{1}{8}&0&\frac{1}{8}&0&0&0&0&0&\frac{1}{4}&0&\frac{1}{4}&0&0&0&0&0&\frac{1}{8}&0&\frac{1}{8}\\
0&0&0&0&\frac{1}{4}&0&0&0&0&0&0&0&\frac{1}{2}&0&0&0&0&0&0&0&\frac{1}{4}&0&0\\
0&0&0&0&0&0&\frac{1}{4}&0&0&0&0&0&0&0&\frac{1}{2}&0&0&0&0&0&0&0&\frac{1}{4}\\
0&0&0&0&0&0&0&0&\frac{1}{2}&0&\frac{1}{2}&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&\frac{1}{2}&0&\frac{1}{2}&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&\frac{1}{2}&0&0&0&0&\frac{1}{2}&0&0&0&0&0&0&0&0&0&0
\end{pmatrix}.$$

[1] A state is said to be recurrent if it will be revisited an infinite number of times in an infinite run of the process. A state is said to be aperiodic if it has a period 1, where the period of a state is the greatest common divisor of the number of times a chain, starting from that state, has a nonzero probability of returning to it.

Let $\boldsymbol{\eta} = (\eta_0 \ \ldots \ \eta_{22})$ be the vector of limiting probabilities of the states of Table 2. We can calculate the values in that vector by solving the following equations for Markov chains

$$\begin{aligned} \boldsymbol{\eta}\mathbf{T} &= \boldsymbol{\eta}, \\ \sum_{j=0}^{22} \eta_j &= 1. \end{aligned} \tag{26}$$

This yields the following

$$\begin{aligned} \boldsymbol{\eta} = (&\frac{13}{1152}, \frac{1}{144}, \frac{43}{2304}, \frac{107}{1152}, \frac{43}{2304}, \frac{1}{144}, \frac{13}{1152}, \frac{13}{2304}, \frac{9}{256}, \frac{91}{1152}, \frac{1}{18}, \\ &\frac{91}{288}, \frac{1}{18}, \frac{91}{1152}, \frac{9}{256}, \frac{13}{2304}, \frac{13}{1152}, \frac{1}{144}, \frac{43}{2304}, \frac{107}{1152}, \frac{43}{2304}, \frac{1}{144}, \frac{13}{1152}). \end{aligned}$$

The average Hamming density of the randomized representation can be obtained by summing the limiting probabilities of the states that have as output $d_i = 1$ or $\overline{1}$.

$$\begin{aligned} Pr(d_i = 1 \text{ or } d_i = \overline{1}) &= \eta_0 + \eta_3 + \eta_6 + \eta_9 + \eta_{10} + \eta_{12} + \eta_{13} + \eta_{16} + \eta_{19} + \eta_{22} \\ &= 0.5 \end{aligned}$$

Similarly, the transition matrix for the states of Table 3, which is for curve E_0, can be formed. By solving (26) for the matrix obtained, the vector of limiting probabilities is found to be

$$\begin{aligned} \boldsymbol{\eta} = (&\frac{1}{144}, \frac{13}{1152}, \frac{43}{2304}, \frac{107}{1152}, \frac{43}{2304}, \frac{13}{1152}, \frac{1}{144}, \frac{13}{2304}, \frac{91}{1152}, \\ &\frac{9}{256}, \frac{1}{18}, \frac{91}{288}, \frac{1}{18}, \frac{9}{256}, \frac{91}{1152}, \frac{13}{2304}, \frac{1}{144}, \frac{13}{1152}, \frac{43}{2304}, \frac{107}{1152}, \frac{43}{2304}, \frac{13}{1152}, \frac{1}{144} \end{aligned}$$

Hence, we have

$$\begin{aligned} Pr(d_i = 1 \text{ or } d_i = \overline{1}) &= \eta_1 + \eta_3 + \eta_5 + \eta_8 + \eta_{10} + \eta_{12} + \eta_{14} + \eta_{17} + \eta_{19} + \eta_{21} \\ &= 0.5 \end{aligned}$$

We can see that for both curves the average Hamming density for the randomized representation is 0.5.

7 Average and Exact Number of Representations

In this section, we first show how to obtain the average number of representations for a τNAF of length l by finding the total number of representations for all τNAFs of length l and dividing it by the number of those τNAFs. Then, we show how the exact number of representations for a τNAF can also be found.

7.1 Number of τNAFs of Length l

We first prove that the number of τNAFs of length l is the integer closest to $2^{l+2}/3$ as was stated by Solinas [16]. That is, this number is

$$\frac{2^{l+2}-1}{3} = \sum_{i=0}^{\frac{l}{2}} 2^{2i}, \qquad \text{for } l \text{ even}, \tag{27}$$

and

$$\frac{2^{l+2}+1}{3} = \sum_{i=0}^{\frac{l+1}{2}} 2^{2i+1} + 1, \qquad \text{for } l \text{ odd}. \tag{28}$$

The number of non adjacent sequences of length l is the number of ways of placing i non-zero symbols in $l+1-i$ possible positions, such that no two non-zero symbols are adjacent, where $0 \le i \le \lceil \frac{l}{2} \rceil$. Each of the i nonzero symbols can be 1 or -1, yielding 2^i choices for their values. Hence, the number of sequences can be expressed as

$$\sum_{i=0}^{\lceil l/2 \rceil} \binom{l+1-i}{i} 2^i. \tag{29}$$

Now we will prove by induction that (29) is equivalent to (27) and (28). It can be easily verified that this is the case for $l=0$ and 1. Now assume that it is true up to some $l=t-1$ where t is even. We will use the following identity [8]

$$\binom{a+1}{e} = \binom{a}{e-1} + \binom{a}{e}, \tag{30}$$

for any real number a and integer e, where by definition

$$\binom{a}{e} = 0 \quad \text{for } e < 0. \tag{31}$$

If a is an integer,

$$\binom{a}{e} = 0 \quad \text{for } e > a. \tag{32}$$

We have

$$\sum_{i=0}^{t/2} \binom{t+1-i}{i} 2^i = \sum_{i=0}^{t/2} \binom{t-i}{i-1} 2^i + \sum_{i=0}^{t/2} \binom{t-i}{i} 2^i. \tag{33}$$

The second term of (33) evaluates to

$$\sum_{i=0}^{\lceil \frac{t-1}{2} \rceil} \binom{(t-1)+1-i}{i} 2^i = \frac{2^{t+1}+1}{3} \tag{34}$$

by using (28).

As for the first term of (33), let $j = i - 1$. Note that the first term of the summation is 0 from (31). Hence, the summation becomes

$$\begin{aligned}
\sum_{j=0}^{t/2} \binom{t-j-1}{j} 2^{j+1} &= 2 \sum_{j=0}^{\frac{t-2}{2}+1} \binom{(t-2)+1-j}{j} 2^j \\
&= 2 \left[\sum_{j=0}^{\frac{t-2}{2}} \binom{(t-2)+1-j}{j} 2^j + \binom{\frac{t}{2}-1}{\frac{t}{2}} 2^{\frac{t}{2}} \right] \\
&= 2 \, [\frac{2^t - 1}{3} + 0] \\
&= \frac{2^{t+1} - 2}{3}, \qquad (35)
\end{aligned}$$

using (27) and (32).

The sum of (35) and (34) yields

$$\sum_{i=0}^{t/2} \binom{t+1-i}{i} 2^i = \frac{2^{t+2} - 1}{3}. \qquad (36)$$

The proof can be similarly carried for t odd. □

One of the reviewers has suggested the following alternative and simpler proof. Let N_l be the number of τNAFs of length l. Then $N_1 = 3$ and $N_2 = 5$. Now, suppose $l \geq 3$. Every length l τNAF can be obtained uniquely as $(u)0$, where (u) is a length $l-1$ τNAF, or as $(u)01$ or $(u)0\bar{1}$, where (u) is a length $l-2$ τNAF. Hence, $N_l = 2N_{l-2} + N_{l-1}$ from which the formula for N_l follows immediately.

7.2 Number of Possible Representations for All τNAFs of Length l

In the following, we will consider the representations of τNAFs on the curve E_1, though the procedure we followed applies to those on the curve E_0. The states of the algorithm in Table 2, together with an initial state s_0 form a nondeterministic finite automaton (NFA) Γ with alphabet $\{\bar{1}, 0, 1\}$ as illustrated in Figure 1. Three directed edges labeled $\bar{1}$, 0 and 1 begin at s_0 and end at s_4, s_{12} and s_{20}, respectively. The final state of Γ is $s12$. Γ accepts the language described by the regular expression $(\varepsilon|1|\bar{1})(0|01|0\bar{1})^*(000)$. This regular expression represents non-adjacent forms when scanned from the least significant end. Three zeros are prepended in order to ensure that the final state s_{12} is reached for any input NAF string as was explained in Section 4.

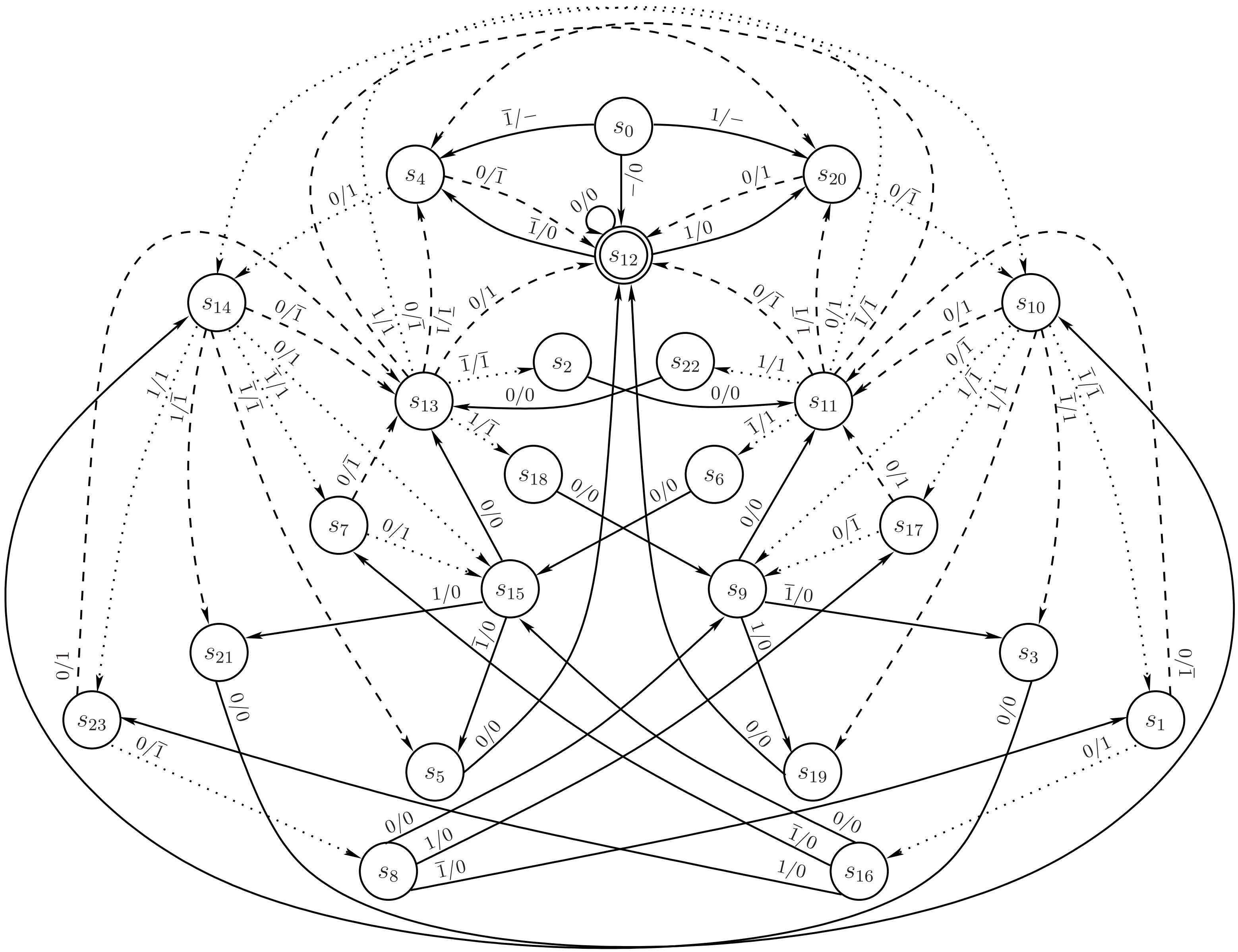

Figure 1: NFA corresponding to Table 2

Since an NFA is a directed graph, it can be described by an *adjacency matrix* $M = (m_{ij})$ for $0 \le i, j \le 23$, such that $m_{ij} = 1$ if there is a directed edge from vertex i to vertex j in Γ and 0 otherwise. The number of directed paths of length l from vertex i to vertex j is the ij-th entry of the matrix M^l.

We can also define an adjacency matrix for each input symbol. For example, M_0 has a 1 in the ij-th entry if there is a directed edge labeled 0 from vertex i to vertex j. Note that since in the automaton considered, starting at some vertex i, there is only one edge labeled with just one of the input symbols that ends at state j, for $0 \le i, j \le 23$, and there are no edges labeled with the empty string ε, we have

$$M = M_{\bar{1}} + M_0 + M_1.$$

Therefore, in order to find all possible paths in Γ for input NAF strings of length l with three prepended 0s, we compute

$$M^l M_0^3 \tag{37}$$

and retrieve its (0,12)th entry. By computing this entry for the different values of l recommended by NIST [12] (163, 233, 283, 409, 571) using MAPLE, we have deduced that it is the integer closest to $1.304812 \cdot 3^l$. The latter along with (27) and (28) gives the average number of representations of a τNAF of length $l \in \{163, 233, 283, 409, 571\}$ as the integer closest to $0.9786 \left(\frac{3}{2}\right)^l$.

The matrix multiplication in (37) can be performed by MAPLE in 0.41 seconds for $l = 163$ and in 0.83 seconds for $l = 571$.

7.3 Exact Number of Representations for a τNAF

The use of adjacency matrices can also be extended to find the number of paths corresponding to a specific input string. That is for a τNAF $k = (\kappa_{l-1}, \ldots, \kappa_1, \kappa_0)_\tau$, the number of possible representations is

$$M_{\kappa_0} M_{\kappa_1} \cdots M_{\kappa_{l-1}} M_0^3 \tag{38}$$

We have included the adjacency matrices for the automaton in Figure 1 in Appendix B.

8 Conclusion

In this article, we have introduced a new method of randomizing the τ-adic representation of a key in ECCs using Koblitz curves. The input to the randomization algorithm is a τNAF of length l. The output of the algorithm is a random τ-adic sequence of the same value as the input. The sbits of the resulting sequence are output one at a time from the least significant to the most significant which allows the simultaneous execution of the scalar multiplication operations. The length of the random representation is at most $l+2$. We have proved that the average Hamming density of all representations for all τNAFs of the same length is 0.5.

We have also presented the pattern of τNAFs with maximum number of representations and the recurrence that governs the number of representations of such τNAFs and have, hence, proved that it is a Fibonacci number and is $\mathcal{O}(\varphi^n)$, where $\varphi \approx 1.618$ is the golden ratio [11].

By modeling our algorithm as a nondeterministic finite automaton and by using adjacency matrices, we have presented a deterministic method to determine the average and the exact number of representations of a τNAF, where the average number is very close to $\left(\frac{3}{2}\right)^l$ for $l \in \{163, 233, 283, 409, 571\}$. It is interesting to note the similarity of the results obtained here to those obtained for the BSD representation of integers [3].

Also of interest is to investigate how this randomization method and the associated properties of the representation can be carried to any complex radix with norm 2 or any arbitrary norm. Note that this complex number should satisfy an equation such as (3), in order to be able to recursively replace digits with a larger absolute value than those in the digit set with the latter ones during the randomization procedure.

References

[1] N. Biggs. *Algebraic graph theory.* Cambridge University Press, 1993. 31

[2] N. Ebeid and A. Hasan. Analysis of DPA countermeasures based on randomizing the binary algorithm. CACR Technical Reports CORR 2003-14, University of Waterloo, 2003. 5

[3] N. Ebeid and M. A. Hasan. On binary signed digit representations of integers. *Designs, Codes and Cryptography*, 42:43–65, 2007. 19, 27

[4] P.-A. Fouque, F. Muller, G. Poupard, and F. Valette. Defeating countermeasures based on randomized BSD representations. In *Cryptographic Hardware and Embedded Systems – CHES '04*, volume 3156 of *LNCS*, pages 312–327. Springer-Verlag, 2004. 12, 13

[5] D. M. Gordon. A survey of fast exponentiation methods. *Journal of Algorithms*, 27(1):129–146, 1998. 14

[6] J. Ha and S. Moon. Randomized signed-scalar multiplication of ECC to resist power attacks. In *Cryptographic Hardware and Embedded Systems – CHES '02*, volume 2523 of *LNCS*, pages 551–563. Springer-Verlag, 2002. 5, 12

[7] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, second edition, 2001. 31

[8] J. G. Kalbfleisch. *Probability and Statistical Inference. Volume 1: Probability.* Springer-Verlag, 1985. 23

[9] D. E. Knuth. *The Art of Computer Programming/Seminumerical Algorithms*, volume 2. Addison-Wesley, second edition, 1973. 2, 12

[10] N. Koblitz. CM curves with good cryptographic properties. In *Advances in Cryptology – CRYPTO '91*, volume 576 of *LNCS*, pages 279–287. Springer-Verlag, 1992. 2

[11] T. Koshy. *Fibonacci and Lucas numbers with Applications.* New York: Wiley, 2001. 18, 27

[12] *National Institute of Standards and Technology. FIPS-186-2: Digital Signature Standard (DSS)*, Jan. 2000. 26

[13] P. J. L. Sang Gyoo Sim, Dong Jin Park. New power analysis on the Ha-Moon algorithm and the MIST algorithm. In *Information and Communications Security – ICICS '04*, volume 3269 of *LNCS*, pages 291–304. Springer-Verlag, 2004. 12

[14] J. Shallit. Personal communication. May, 2006. 31

[15] M. Sipser. *Introduction to the theory of computation.* Boston : PWS Pub. Co, 1997. 31

[16] J. A. Solinas. Efficient arithmetic on Koblitz curves. *Designs, Codes and Cryptography*, 19:195–249, 2000. 2, 3, 4, 5, 7, 21, 23

[17] W. M. O. Staffelbach. Efficient multiplication on certain nonsupersingular elliptic curves. In *Advances in Cryptology – CRYPTO '92*, volume 740 of *LNCS*, pages 333–344. Springer-Verlag, 1993. 13

Appendix

A Examples

Examples of Representations

The following tables present the different representations of the “positive” τNAFs on the curve E_0 and their number.

Table 4: Representations of “positive” τNAFs of length 1.

τNAF k	Representations	$\vartheta(k,1)$
0	0	1
1	1, $\overline{111}$	2

Table 5: Representations of “positive” τNAFs of length 2.

τNAF k	Representations	$\vartheta(k,2)$
0	0	1
1	1, $\overline{111}$, $101\bar{1}$	3
10	10, $\overline{111}0$	2

Table 6: Representations of “positive” τNAFs of length 3.

τNAF k	Representations	$\vartheta(k,3)$
0	0	1
1	1, $\overline{111}$, $111\overline{11}$, $101\bar{1}$	4
10	10, $\overline{111}0$, $101\bar{1}0$	3
$10\bar{1}$	$10\bar{1}$, $\overline{111}0\bar{1}$, $\overline{11}011$, $\bar{1}1\bar{1}1$	4
100	100, $\overline{111}00$	2
101	101, $\overline{111}01$, $\overline{11}$, $111\bar{1}$, $\bar{1}0\bar{1}1\bar{1}$	5

Table 7: Representations of “positive” τNAFs of length 4.

τNAF k	Representations	$\vartheta(k, 4)$
0	0	1
1	1, $\bar{1}\bar{1}\bar{1}$, $111\bar{1}\bar{1}$, $\bar{1}0\bar{1}1\bar{1}\bar{1}$, $101\bar{1}$, $\bar{1}\bar{1}\bar{1}01\bar{1}$	6
10	10, $\bar{1}\bar{1}\bar{1}0$, $111\bar{1}\bar{1}0$, $101\bar{1}0$	4
$10\bar{1}$	$10\bar{1}$, $\bar{1}\bar{1}\bar{1}0\bar{1}$, $101\bar{1}0\bar{1}$, $\bar{1}\bar{1}011$, 101011, $\bar{1}1\bar{1}1$, $1111\bar{1}1$, $100\bar{1}\bar{1}1$	8
100	100, $\bar{1}\bar{1}\bar{1}00$, $101\bar{1}00$	3
101	101, $\bar{1}\bar{1}\bar{1}01$, $101\bar{1}01$, $\bar{1}\bar{1}$, $111\bar{1}$, $\bar{1}\bar{1}\bar{1}11\bar{1}$, $\bar{1}0\bar{1}1\bar{1}$	7
$10\bar{1}0$	$10\bar{1}0$, $\bar{1}\bar{1}\bar{1}0\bar{1}0$, $\bar{1}\bar{1}0110$, $\bar{1}1\bar{1}10$	4
$100\bar{1}$	$100\bar{1}$, $\bar{1}\bar{1}\bar{1}00\bar{1}$, 1111, $\bar{1}\bar{1}\bar{1}111$, $\bar{1}0\bar{1}11$, $\bar{1}1$	6
1000	1000, $\bar{1}\bar{1}\bar{1}000$	2
1001	1001, $\bar{1}\bar{1}\bar{1}001$, $1\bar{1}\bar{1}\bar{1}$, $\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}$, $\bar{1}001\bar{1}\bar{1}$, $\bar{1}\bar{1}001\bar{1}$	6
1010	1010, $\bar{1}\bar{1}\bar{1}010$, $\bar{1}\bar{1}0$, $111\bar{1}0$, $\bar{1}0\bar{1}1\bar{1}0$	5

Examples of $k_{max,l}$

Table 8 presents $k_{max,l}$ and $\vartheta(k_{max,l}, l)$ for $1 \leq l \leq 13$.

Table 8: “Positive” τNAFs with maximum number of representations

l	$k_{max,l}$	$\vartheta(k_{max,l}, l)$
1	1	2
2	1	3
3	101	5
4	$10\bar{1}$	8
5	$1010\bar{1}$	13
6	$10\bar{1}01$	21
7	$1010\bar{1}01$	34
8	$10\bar{1}010\bar{1}$	55
9	$1010\bar{1}010\bar{1}$	89
10	$10\bar{1}010\bar{1}01$	144
11	$1010\bar{1}010\bar{1}01$	233
12	$10\bar{1}010\bar{1}010\bar{1}$	377
13	$1010\bar{1}010\bar{1}010\bar{1}$	610

B Nondeterministic Finite Automata, Directed Graphs and Adjacency Matrices

A *nondeterministic finite automaton* (NFA) Γ is a quintuple $(Q, \Sigma, s_0, F, \delta)$ [7], where

- Q is a set of *states*,
- Σ is the alphabet (set) of *input symbols*,
- $s_0 \in Q$ is the *initial state*,
- $F \subset Q$ is the set of *final* (or *accepting*) *states*,
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the *transition function*, where $\mathcal{P}(Q)$ is the *powerset* of Q, that is, the set of all subsets of Q (including the empty set).

Let X be a string over the alphabet Σ, and ε be the empty string. Γ *accepts* the string X if there exist both a representation of X of the form $x_1x_2\ldots x_l$, $x_i \in (\Sigma \cup \{\varepsilon\})$, and a sequence of states $s_0, s_1, \ldots, s_l$, $s_i \in Q$, meeting the following conditions:

- s_0 is the initial state,
- $s_i \in \delta(s_{i-1}, x_i)$, for $1 \le i \le l$ and
- $s_l \in F$. [15, Section 1.2, pp.47-63]

An NFA can be represented by a directed graph where the vertices are the states of the set Q, and the directed edges are determined by the function δ. That is, a directed edge exists starting at vertex s_i and ending at vertex s_j iff $s_j \in \delta(s_i, x)$, for any $x \in \Sigma$, and this edge will be labeled as x. The concatenation of directed edges encountered when Γ is reading an accepted string form a *directed path*.

To each directed graph, we can associate the *adjacency matrix*, $M = (m_{ij})$ for $0 \le i, j \le |Q|$, such that $m_{ij} = 1$ if there is a directed edge from vertex s_i to vertex s_j in Γ and 0 otherwise. From the definition of matrix multiplication and the concatenation of paths, the l^{th} power of M, *i.e.*, M^l has the number of paths of length l from vertex s_i to vertex s_j as its ij^{th} entry. This is obviously true for $l = 1$. Next observe that any path of length l from vertex s_i to vertex s_j decomposes into the initial path of length $l-1$ starting at s_i (to some intermediate vertex) followed by a path of length 1 ending at s_j, these paths are counted for all possible intermediate vertices by the sum of the vector product of the i^{th} row of M^{l-1} with the j^{th} column of M [1, Lemma 2.5].

Moreover, to an NFA Γ, we can associate an adjacency matrix, M_{x_i}, for each input symbol $x_i \in \Sigma$, $1 \le i \le |\Sigma|$. Hence the number of directed paths possibly traversed when Γ reads an accepted string $X = x_1x_2\ldots x_l$ can be found as the $(0, f)^{th}$ entry of the product [14]

$$M_{x_1}M_{x_2}\cdots M_{x_l},$$

for each $f \in F$ possibly reached when x_l was read.

The following are the adjacency matrices corresponding to the automaton in Figure 1.

$$M = \begin{pmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

$$M_{\bar{1}} = \begin{pmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

$$
M_0 = \begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0
\end{pmatrix}
$$

$$
M_1 = \begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix}
$$