A fast and SCA-resistant algorithm for exponentiation using random Euclidean addition chain and the constrained discrete logarithm problem

Fabien Herbaut, Nicolas Méloni, and Pascal Véron

1 Université du Sud Toulon-Var, IMATH, France
IUFM de Nice, Université de Nice
herbaut@unice.fr

2 Department of Electrical and Computer Engineering, Waterloo University, Canada
nmeloni@vlsi.uwaterloo.ca

3 Université du Sud Toulon-Var, IMATH, France
veron@univ-tln.fr

Abstract. Efficiency and security are the two main objectives of every elliptic curve scalar multiplication implementations. Many schemes have been proposed in order to speed up or secure its computation, usually thanks to efficient scalar representation [1–3], faster point operation formulae [4–6] or new curve shapes [7]. In [8, 9], authors suggest exponentiation algorithms that are faster than average for exponents sampled from certain subsets. In this paper, we use a similar approach. We propose to directly modify the key generation process in order to produce small Euclidean chains instead of scalar. This allows us to use a previous scheme, secure against side channel attacks, but whose efficiency relies on the computation of small chains computing the scalar. We propose two different ways to generate short Euclidean chains and give a first theoretical analysis of the size and distribution of the obtained keys. We also propose a new scheme in the context of fixed base point scalar multiplication.

1 Introduction

After 25 five years of existence, elliptic curve cryptography (ECC) is now one of the major public-key cryptographic primitives. Its main advantages, compared to its main competitor RSA, are its shorter keys and the lack of fast theoretical attacks. The recent factorization of an RSA modulus of 768 bits [10] is here to highlight the significant role that ECC will play during the next decade. In particular, it has been shown that ECC is suitable for cryptographic applications on devices with small resources. However, if 160-bit ECC is believed to remain secure, from a theoretical point of view, at least until 2020 [11], physical attacks on cryptographic devices have proved to be an immediate threat [12]. Thus, software or hardware ECC implementations have to deal with two apparently opposite requirements: efficiency and security. Indeed, protecting a device from physical attacks usually involves costly countermeasures.

In this work, we take a different approach in order to propose a fast and secure scalar multiplication scheme. In 2007, Meloni proposed a secure algorithm based on Euclidean addition chains [13]. As they only involve additions, they are naturally resistant to SCA. However, the efficiency of such a method relies on the existence of a small chain computing the scalar. It has been pointed out that finding such a chain becomes more and more difficult when the scalar grow in size. For cryptographic sizes, finding a good chain is costlier than the scalar multiplication itself. So, instead of proposing a new scalar multiplication scheme, we propose to directly modify the key generation process. More precisely, we show that it is possible to generate the key as a small Euclidean chain, allowing us to use Meloni’s fast and secure scheme. Yet, this approach arises several problems. First, what is the size of the keys
we can generate for a given chain length? In other words, is it possible to achieve a certain level of security with relatively small chains? The second problem is that of distribution. Indeed, many different chains of same length can compute the same integer. It is then important to ensure that generating keys this way does not weaken the discrete logarithm problem.

In this paper, we produce the first practical and theoretical results on random Euclidean chain generation. We also show that it can lead to efficient and SPA-secure scalar multiplication methods.

This work is organized as follows. In Section 2 we give some recalls on Meloni’s scheme. In Section 3 we recall some background about Euclidean chains and set notations. In Section 4 and 5, we describe two different families of Euclidean addition chains and give some results on their distribution (notice that in Section 4 two variants are described). Finally, in Section 6 we propose some comparisons with existing side-channel resistant scalar multiplication methods.

2 Scalar multiplication using Euclidean addition chains

For the sake of concision, we do not give any recalls about scalar multiplication and side channel attacks. We invite the reader to refer to [14, 15] for detailed overview of elliptic curve based cryptography.

This section is dedicated to a specific scalar multiplication algorithm based on Euclidean addition chains.

2.1 The z coordinate trick

Let $E : Y^2 = X^3 + aXZ^4 + bZ^6$ be an elliptic curve defined over a prime field $K$. Assume that the points are given in Jacobian coordinates ($\mathbf{P}(X,Y,Z) \sim (\lambda^2 X, \lambda^3 Y, \lambda Z)$). Let now $P_1$ and $P_2$ be two points sharing the same $z$-coordinate $Z$, $P_1 + P_2$ can be obtained using the following formulae:

$$P_1 = (X_1, Y_1, Z), \quad P_2 = (X_2, Y_2, Z) \quad \text{and} \quad P_1 + P_2 = (X_3, Y_3, Z_3)$$

$$A = (X_2 - X_1)^2, \quad B = X_1 A, \quad C = X_2 A, \quad D = (Y_2 - Y_1)^2$$

and

$$X_3 = D - B - C, \quad Y_3 = (Y_2 - Y_1)(B - Y_3) - Y_1(C - B), \quad Z_3 = Z(X_2 - X_1).$$

This addition involves 5 field multiplications (M) and 2 squaring (S), which is faster than any other group operation on the curves. Another interesting feature is that the quantities $X_1 A = X_1 (X_2 - X_1)^2$ and $Y_1 (C - B) = Y_1 (X_2 - X_1)^3$ computed during the point addition can be seen as the $x$ and $y$-coordinates of the point $(X_1 (X_2 - X_1)^2, Y_1 (X_2 - X_1)^3, Z(X_2 - X_1)) \sim (X_1, Y_1, Z)$. Thus it is possible to add $P_1$ and $P_1 + P_2$ with the same formulae. Said differently, we have a function $ZADD$ that takes as input two points $P_1$ and $P_2$ sharing the same $z$-coordinate and returns the two points $P_1 + P_2$ and $P_1$ sharing the same $z$-coordinate. Moreover, due to the symmetry of point addition, we can see that $ZADD(P_2, P_1) = (P_2 + P_1, P_3)$. This means that for each addition, it is possible to choose whether $P_1$ or $P_2$ will be used for the next addition.

In [13], it is shown that any scalar multiplication can be performed using the $ZADD$ function only. It is achieved by finding an Euclidean addition chain computing the scalar.
2.2 Euclidean addition chains

Definition 1. An Euclidean addition chain (EAC) of length \( s \) is a sequence \( (c_i)_{i=1}^s \) with \( c_i \in \{ 0, 1 \} \). The integer \( k \) computed from this sequence is obtained from the sequence \( (v_i, u_i)_{i=0}^s \) such that \( v_0 = 1, u_0 = 2 \) and \( \forall i \geq 1, (v_i, u_i) = (v_{i-1}, v_{i-1} + u_{i-1}) \) if \( c_i = 1 \) (small step), or \( (v_i, u_i) = (u_{i-1}, v_{i-1} + u_{i-1}) \) if \( c_i = 0 \) (big step). The integer \( k \) associated to the sequence \( (c_i)_{i=1}^s \) is \( v_s + u_s \).

Example : From the EAC \((10110)\) one can compute the integer 23 as follows : \( (1, 2) \rightarrow (1, 3) \rightarrow (3, 4) \rightarrow (3, 7) \rightarrow (3, 10) \rightarrow (10, 13) \rightarrow 23 \).

One can perform a scalar multiplication using Algorithm 1.

<table>
<thead>
<tr>
<th>Algorithm 1 Scalar multiplication using ZADD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Require</strong>: ( P, [2]P ) with ( ZP = Z_{[2]P} ) and an EAC ( c = (c_1, \ldots, c_s) ) computing ( k )</td>
</tr>
<tr>
<td><strong>Ensure</strong>: ( U_1 = kP )</td>
</tr>
</tbody>
</table>

1: \( (U_1, U_2) \leftarrow ([2]P, P) \)
2: for \( i = 1 \ldots s \) do
3: if \( c_i = 0 \) then
4: \( (U_1, U_2) \leftarrow \text{ZADD}(U_1, U_2) \)
5: else
6: \( (U_1, U_2) \leftarrow \text{ZADD}(U_2, U_1) \)
7: end if
8: \( (U_1, U_2) \leftarrow \text{ZADD}(U_1, U_2) \)
9: end for
10: return \( U_1 \)

As stated in [13] the total computational cost of Algorithm 1 is \((5s - 7)M + (2s - 1)S\). As shown in the original paper, some cryptographic protocols only require the \( x \)-coordinate of the point \( kP \). In this case, it is possible to save one multiplication by step of Algorithm 1 by noticing that \( Z \) does not appear during the computation of \( X_3 \) and \( Y_3 \), thus it is not necessary to compute \( Z_3 \) during the process. In the end, it is still possible to recover the \( x \)-affine-coordinate.

The efficiency of this method directly depends on the length of the chains. We can also remark that, at each step of the for loop, there are only two possible operations. Either \( \text{ZADD}(U_1, U_2) \) or \( \text{ZADD}(U_2, U_1) \) is computed, which means that the same operation is used every time. This property makes the algorithm resistant to side simple channel attacks. Differential attacks can be simply countered using randomization of the coordinates of the base point, which only adds 3 multiplications and one squaring to the overall cost.

Although finding an Euclidean chain computing a given integer \( k \) is quite simple (it suffices to choose an integer \( g \) co-prime with \( k \) and apply the subtractive form of Euclid’s algorithm) finding a short chain remains a hard problem. As an example, the average length of the computed chain for \( k \) with \( g \) uniformly distributed in the range \([1, k]\) is \( O(\ln k)^2 \) [16]. For 160-bit scalars, experiments have shown that, on average, it is required to try more than 45,000 different \( g \) to find a relatively small chain using the Montgomery heuristic [17].

We propose in this paper to proceed differently. Instead of randomly choosing an integer \( k \) and then trying to find a suitable EAC \( c \) to finally compute the point \( kP \), we propose to randomly generate a small length EAC \( c \) and then compute the associated point on the curve.

Using this approach we compute points \( kP \) for a subset \( S \) of all possible values for the integers \( k \). Hence we do not deal any more with the classical Discrete Logarithm Problem but with the Constrained Discrete Logarithm Problem.
3 Notations and Properties

We give in this section some notations and important results for the sequel of this paper.

Definition 2. Let $n$ and $p$ be two integers, we define:

- $\mathcal{M}$ as the set of finite EAC,
- $\mathcal{M}_n$ as the set of EAC of length $n > 0$,
- $\mathcal{M}_{n,p}$ as the set of EAC of length $n > 0$ and Hamming weight $p > 0$.
- $\chi$ the map from $\mathcal{M}$ to $\mathbb{N}$, such that for $m \in \mathcal{M}$, $\chi(m)$ be the integer computed from the EAC $m$,
- $\psi$ the map from $\mathcal{M}$ to $\mathbb{N} \times \mathbb{N}$, such that for $m \in \mathcal{M}$, $\psi(m) = (v_s, u_s)$ if $m \in \mathcal{M}_s$,
- $S_0$ the matrix $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ corresponding to a big step iteration,
- $S_1$ the matrix $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ corresponding to a small step iteration.

With these notations, for $m = (m_1, \ldots, m_s) \in \mathcal{M}_s$, we have:

$$\psi(m) = (1, 2) \prod_{i=1}^s S_{m_i} \text{ and } \chi(m) = ((1, 2) \prod_{i=1}^s S_{m_i}, (1, 1)).$$

Let $r$ and $s$ be two integers, we will denote by $mm'$ the element of $\mathcal{M}_{r+s}$ obtained from the concatenation of $m \in \mathcal{M}_r$ and $m' \in \mathcal{M}_s$. This way, for $n > 0$, $m^n$ is a word of $\mathcal{M}_{nr}$ if $m \in \mathcal{M}_r$.

For convenience, $\mathcal{M}_0$ will correspond to the set with one element $e$ which is the identity element for the concatenation.

For $m$ and $m'$ two elements of $\mathcal{M}_r$ such that $\psi(m) = (v, u)$ and $\psi(m') = (v', u')$ we will say that $\psi(m) \leq \psi(m')$ if $v \leq v'$ and $u \leq u'$.

Proposition 1. Let $n > 0$, $F_i$ be the $i$th Fibonacci number, $\alpha_n = \frac{(1+\sqrt{2})^n + (1-\sqrt{2})^n}{2\sqrt{2}}$ and $\beta_n = \frac{(1+\sqrt{2})^n - (1-\sqrt{2})^n}{2\sqrt{2}}$:

- $\psi(0^n) = (F_{n+2}, F_{n+3})$, $\psi(1^n) = (1, n+2)$, $\chi(0^n) = F_{n+4}$, $\chi(1^n) = n+3$,
- $\forall m \in \mathcal{M}_n$, $\chi(1^n) \leq \chi(m) \leq \chi(0^n)$, and $\psi(1^n) \leq \psi(m) \leq \psi(0^n)$,
- $S_0^n = \begin{pmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{pmatrix}$, $S_1^n = \begin{pmatrix} 1 & n \\ 0 & 1 \end{pmatrix}$,
- $(S_0 S_1)^n = \begin{pmatrix} \alpha_n - \beta_n & \beta_n \\ \beta_n & \alpha_n + \beta_n \end{pmatrix}$, $(S_1 S_0)^n = \begin{pmatrix} \alpha_n & 2\beta_n \\ \beta_n & \alpha_n \end{pmatrix}$.

Proof. The first property is straightforward. The other ones can be proved by induction.
Notice that from $M_{0}^{m+n} = M_{0}^{m}M_{0}^{n}$, we can find back a well known identity on Fibonacci sequence, namely:

$$F_{m+n} = F_{m-1}F_{n} + F_{m}F_{n+1}. \quad (1)$$

**Proposition 2.** Let $n > 0$ and $m = (m_{1}, \ldots, m_{n}) \in \mathcal{M}_{n}$, then:

1. $\chi(m_{1}, \ldots, m_{n}) = \chi(m_{n}, \ldots, m_{1})$,
2. the map $\psi$ is injective.

**Proof.** We refer to [19] for standard link between EAC, Euclidean algorithm and continued fractions, which explains the first point. It is also explained that if $\psi(m) = (v, u)$ then $(u, v) = 1$ and the only chain which leads to $(v, u)$ is obtained using the additive version of Euclidean algorithm.

### 4 A first family of EAC

We will consider in this section $\mathcal{M}_{n}^{0}$ the subset of $\mathcal{M}_{2n}$ whose elements are EAC beginning with $n$ zeros.

#### 4.1 Some properties of $\mathcal{M}_{n}^{0}$

From proposition 2 the restriction of $\chi$ to $\mathcal{M}_{n}$ is not injective because of the mirror symmetry property.

**Proposition 3.** The restriction of $\chi$ to $\mathcal{M}_{n}^{0}$ is injective.

**Proof.** Let $x$ and $y$ be two words of $\mathcal{M}_{n}^{0}$ such that $\chi(x) = \chi(y)$, and $m0^n$, $m'0^n$, be the words obtained when reading $x$ and $y$ from right to left. Using the symmetry property, we have $\chi(m0^n) = \chi(m'0^n)$. Let $(v, u) = \psi(m)$ and $(v', u') = \psi(m')$, then

$$\chi(m0^n) = \chi(m'0^n) \iff F_{u}v + F_{n+1}u + F_{n}v = F_{u'}v' + F_{n-1}u' + F_{n+1}u' + F_{n}v' \iff F_{n+2}(u - u') = F_{n+1}(v' - v)$$

Since $(F_{n+1}, F_{n+2}) = 1$, then $F_{n+2}$ divides $v' - v$. Now from proposition 1, since $v$ and $v'$ are less or equal than $F_{n+2}$ and nonzero, then $|v' - v| < F_{n+2}$ which implies that $v = v'$ and so $u = u'$. Hence $\psi(m) = \psi(m')$, so $m = m'$.

**Proposition 4.** $\chi(\mathcal{M}_{n}^{0}) \subset [(n + 1)F_{n+2} + F_{n+3}, F_{2n+4}]$, the lower (resp. the upper) bound being reached by $0^{n+1}n$ (resp. $0^{2n}n$). The mean value is $(\frac{1}{2})^n F_{n+4}$. 

**Proof.** Let $0^a x 1 y$ and $0^b x 0 y$ be two elements of $\mathcal{M}_{n}^{0}$ where $x$ and $y$ are chains of size $a$ and $n - 1 - a$ with $a \in [0, n - 1]$. From the definition of $\chi$ it follows that $\chi(0^a x 1 y) < \chi(0^b x 0 y)$. Hence the smallest integer is computed from the word $0^n 1^n$ and the greatest from $0^n 0^n$. Now $\chi(0^n 1^n) = ((1, 2)S_{0}^{n}S_{0}^{n}, (1, 1))$ and $\chi(0^{2n}) = ((1, 2)S_{0}^{2n}, (1, 1))$. From proposition 1, we deduce that $\chi(0^n 1^n) = (n + 1)F_{n+2} + F_{n+3}$ and $\chi(0^{2n}) = F_{2n+4}$.

To compute the mean value, let us consider $n$ independent Bernoulli random variables $C_{1}, \ldots, C_{n}$ such that $\forall i \in [1, n]$, $\Pr(C_{i} = 0) = \Pr(C_{i} = 1) = 1/2$. The mean value is $E(X)$ where $X = \chi(0^n 1^1 \ldots 1^1)$. Now

$$X = \langle (1, 2)S_{0}^{n} \prod_{i=1}^{n} (\frac{C_{i}}{1-C_{i}}), (1, 1) \rangle.$$

Notice that $X$ is a polynomial of $Z[C_{1}, \ldots, C_{n}]/(C_{1}^2 - C_{1}, \ldots, C_{n}^2 - C_{n})$. As the $C_{i}$ are independents then $\forall J \subset [1, n]$, $E(\prod_{i \notin J} C_{i}) = \prod_{i \notin J} E(C_{i})$, hence

$$E(X) = ((1, 2)S_{0}^{n} \prod_{i=1}^{n} (\frac{E(C_{i})}{1-E(C_{i})}), (1, 1)) = ((1, 2)S_{0}^{n} \prod_{i=1}^{n} (\frac{1}{2}), (1, 1)).$$
The final result comes from proposition 1 and the equality:
\[
\forall n \in \mathbb{N}^*, \left(\frac{\psi + 1}{\psi - 1}\right)^n = (3/2)^{n-1}\left(\frac{\psi + 1}{\psi - 1}\right).
\]

4.2 Application to existing standards

Using the set \( \chi(\mathcal{M}_{160}^0) \), we can generate (with algorithm 1) \( 2^n \) distinct points \( \chi(c)P \) for a point \( P \) whose order is greater than \( F_{2n+4} \). Of course, when the order \( d \) of the point \( P \) is known, we have to choose the largest integer \( n \) such that \( F_{2n+4} < d \).

Because of the results on the difficulty to solve the CDLP problem, we have to consider the set \( \mathcal{M}_{160}^0 \) only for \( n \geq 160 \). For \( n = 160 \), we have \( \chi(\mathcal{M}_{160}^0) \subset \{161F_{162} + F_{163}, F_{324}\} \), that is to say
\[
\chi(\mathcal{M}_{160}^0) \subset [2^{118.6} \cdot 2^{223.6}].
\]

Such data fit well with the secp224k1 and secp224r1 parameters where the order of the point \( P \) is about \( 2^{223.99} \) [20]. Those parameters are consistent with ANSI X.962, IEEE P1363 and IPSec standards and are recommended for ANSI X9.63 and NIST standards.

4.3 A variant for the secp160k1 and secp160r1 recommended parameters

In the secp160k1 and secp160r1 recommended parameters, the order \( d \) of the point \( P \) is around \( 2^{160} \). Using the set \( \mathcal{M}_{160}^0 \), this leads us to choose \( n = 114 \) which only gives rise to \( 2^{114} \) distinct points.

Notice that if we use an element \( c \) of \( \mathcal{M}_{160}^0 \) with the point \( P \) of order \( d \) then the algorithm 1 computes \( \chi(c) \mod d)P \). If the values of \( \{\chi((c) \mod d)P, c \in \mathcal{M}_{160}^0\} \) are well distributed among \( \mathbb{Z}/d\mathbb{Z} \), then we can use the above mentioned method, provided that computing \( \chi(c)P \) with algorithm 1 be more efficient than computing \( kP \) with \( k \in \mathbb{Z}/d\mathbb{Z} \), with a classical SPA-resistant method. This last point will be discussed in section 6, we will focus now on the problem of the distribution. To this end, we will adapt results on Stern sequences from [21].

**Definition 3.** Let \((a, b) \in \mathbb{N}^2\), the generalized Stern sequence \((s_{a,b}(r, n))_{r \in \mathbb{N}, n \in [0, 2^r]}\) is defined by \( s_{a,b}(0, 0) = a, s_{a,b}(0, 1) = b \), and for \( r \geq 1 \), \( s_{a,b}(r, 2n) = s_{a,b}(r − 1, n), s_{a,b}(r, 2n + 1) = s_{a,b}(r − 1, n) + s_{a,b}(r − 1, n + 1) \).

In his original paper, Stern gave a practical description of his sequence using the following diatomic array [22]:

\[
\begin{array}{cccc}
(r = 0) & a & b \\
(r = 1) & a & a + b & b \\
(r = 2) & a & 2a + b & a + b & a + 2b & b \\
(r = 3) & a & 3a + b & 2a + 3a & 2a + 2b + a & 3b + 2b + a + 3b & b \\
\vdots
\end{array}
\]

where each line \( r \) is exactly the sequence \( s_{a,b}(r, n) \) for \( n \in [0, 2^r] \). Notice that to compute the row \( r \), you just have to rewrite row \( r − 1 \) and insert between two elements their sum. In the case \((a, b) = (1, 1)\), the sequence is called the Stern sequence and has been well studied.

For example, see the introduction of [21] or [23] for the link with the Stern Brocot array.

Now we will point deep connections between Stern sequences and the \( \psi \) and \( \chi \) maps. These connections should not surprise us, because both are linked with continued fractions. As an
example, if \((a, b) = (1, 1)\), an easy induction enables us to prove that when \(n \in [0, 2^r]\), the sequence \((s_{1, 1}(r + 1, 2n), s_{1, 1}(r + 1, 2n + 1))\) describes the set \(\psi(\mathcal{M}_r)\).

\[
\begin{align*}
(r = 0) & : 1 \ 1 \\
(r = 1) & : 1 \ 2 \ 1 \\
(r = 2) & : 1 \ 3 \ 2 \ 3 \ 1 \\
(r = 3) & : 1 \ 4 \ 3 \ 5 \ 2 \ 5 \ 3 \ 4 \ 1 \\
& \vdots
\end{align*}
\]

Let us note \(\Delta : \mathbb{N}^2 \to \mathbb{N}^2\) such that \(\Delta(x, y) = (y, x)\). Another induction enables to prove that when \(n \in [0, 2^{r+1} - 1]\), \((s_{1, 1}(r + 1, n), s_{1, 1}(r + 1, n + 1))\) describes \(\psi(\mathcal{M}_r) \cup \Delta(\psi(\mathcal{M}_r))\).

For \(\ell \in \mathbb{N}^*\) and \(m \in \mathcal{M}_\ell\), let \(A_r(m) = \{\psi(mx) \mid x \in \mathcal{M}_r\} \cup \{\Delta(\psi(mx)) \mid x \in \mathcal{M}_r\}\).

From now on, in order to simplify the notations, we will denote by \(s(r, n)\) the value \(s_{a,b}(r, n)\). Let us define the sequences \((S(r, n))_{r \in \mathbb{N}, n \in [0, 2^r - 1]}\) by \(S(r, n) = (s(r, n), s(r, n + 1))\) and \((S_d(r, n))_{r \in \mathbb{N}, n \in [0, 2^r - 1]}\) by \(S_d(r, n) = (s(r, n) \mod d, s(r, n + 1) \mod d)\). The following link between \(\psi\) and \(S\) can be proved by induction.

**Lemma 1.** Let \(r \geq 0\), \(\ell > 0\) and let \(m \in \mathcal{M}_\ell\) such that \(\psi(m) = (a, b)\). Then \(S(r + 1, \ldots)\) is a one to one map from \([0, 2^{r+1} - 1]\) onto \(A_r(m)\).

It means that in our case, the values \(\psi(c)\) and \(\Delta(\psi(c))\) for \(c \in \mathcal{M}_\ell^d\) correspond to the elements \(S(\ell + 1, n)\) for \(n \in [0, 2^{\ell+1} - 1]\) and \((a, b) = (F_{\ell+2}, F_{\ell+3})\). Recently, Reznick proved in [21] that, for \(d \geq 2\), \((a, b) = (1, 1)\), and \(r\) sufficiently large, the sequence \((S_d(r, n))_{n \in \mathbb{N}}\) is well distributed among \(S_d := \{(i \mod d, j \mod d) \mid \gcd(i, j, d) = 1\}\). We need a similar result for any couple \((a, b)\) in order to show that the values \(\chi(c)\) are asymptotically well distributed modulo \(d\). We will use similar notations and follow the arguments of [21] to prove the next theorem.

We define:

- for \(\gamma \in S_d\), \(B_d(r, \gamma) := \#\{n \in [0, 2^r - 1] \mid S_d(r, n) = \gamma\}\),
- \(\chi_d\), the map such that \(\chi_d(m) = \chi(m) \mod d\),
- \(\psi_d\), the map such that if \(\psi(m) = (v, u)\) then \(\psi_d(m) = (v \mod d, u \mod d)\),
- \(N_d\) the cardinality of \(S_d\).

**Theorem 1.** Let \((a, b, d) \in \mathbb{N}^3\) such that \(d\) be prime and \((a, d) = (b, d) = 1\). There exist constants \(c_d\) and \(\rho_d < 2\) so that if \(m \in \mathbb{N}\) and \(\alpha \in S_d\), then for all \(r \geq 0\),

\[
|B_d(r, \alpha) - 2^r/N_d| < c_d\rho_d^r.
\]

**Proof.** Due to the lack of space, we just give a short proof of this, following the arguments of section 4 in [21] and pointing out the differences. Since \(d\) is prime, we have \(S_d = \mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}/d\mathbb{Z} \setminus \{(0, 0)\}\), and \(N_d = d^2 - 1\). We can define a graph \(G_d\) and the applications \(L\) and \(R\) in the same way, and also have \(L^d = R^d = \text{id}\). In the proof of lemma 14, we have to give a slightly different proof that for each \(\alpha = (x, y) \in S_d\) there exists a way from \((a, b)\) to \(\alpha\) in the graph \(G_d\). Notice that since \((0, 0) \notin S_d\), then either \(x \neq 0\) or \(y \neq 0\). If \(y \neq 0\), we notice that \(R^d(L^d(a, b)) = (a + k'(b + ka), b + ka)\). As \((a, d) = 1\), we can choose \(k\) such that \(b + ka = y\). Thus, as \(y \neq 0\), we can choose \(k'\) such that \(a + k'(b + ka) = x\) and we are done. If \(x \neq 0\), then we consider \(L^d(R^d(a, b)) = (a + kb, b + k'(a + kb))\) : in the same way, we can choose \((k', k)\) such that \(a + kb = x\) and \(b + k'(a + kb) = y\). In the first line of the proof of lemma 14, we also have to consider \((r_0, v_0) \in \mathbb{N} \times \mathbb{N}\) such that \(S_d(r_0, v_0) = (0, 1)\) rather than \(S_d(0, 0) = (0, 1)\). Thus the adjacency matrix of the graph satisfies the same properties as in theorem 15 of [21]. So the conclusion remains true for \(B_d\).
We can deduce from Theorem 1 and Lemma 1 an interesting regularity property for $\chi_d(mx)$ when $m$ is fixed and $x$ lies in $\mathcal{M}_r$.

**Theorem 2.** Let $m \in \mathcal{M}_r$, there exist constants $c_d \in \mathbb{R}_+$ and $\tau_d \in [0,1]$ so that for all $\alpha \in (\mathbb{Z}/d\mathbb{Z})^*$, $r \in \mathbb{N}$

$$\begin{align*}
\left|\frac{\#\{x \in \mathcal{M}_r \mid \chi_d(mx) = \alpha\}}{2^r} - \frac{d}{d^2 - 1}\right| < c_d \tau_d^r, \quad \text{and} \\
\left|\frac{\#\{x \in \mathcal{M}_r \mid \chi_d(mx) = 0\}}{2^r} - \frac{1}{d+1}\right| < c_d \tau_d^r.
\end{align*}$$

**Proof.** Let $\alpha \in (\mathbb{Z}/d\mathbb{Z})^*$, and $(\beta_i, \gamma_i)_{1 \leq i \leq d}$ the $d$ elements of $\mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}/d\mathbb{Z} \setminus \{(0,0)\}$ such that $\beta_i + \gamma_i = \alpha$, then

$$\#\{x \in \mathcal{M}_r \mid \chi_d(mx) = \alpha\} = \sum_{i=1}^{d} \#\{x \in \mathcal{M}_r \mid \psi_d(mx) = (\beta_i, \gamma_i)\}.$$ 

If $n \in [0,2^{r+1}]$ is such that $S_d(r+1,n) = (\beta_i, \gamma_i)$ then, thanks to Lemma 1, it corresponds to $x \in \mathcal{M}_r$ such that $\psi_d(mx) = (\beta_i, \gamma_i)$ or $\psi_d(mx) = (\gamma_i, \beta_i)$. In this last case, there exists an integer $j$, such that $(\gamma_i, \beta_j) = (\beta_j, \gamma_j)$.

Thus,

$$\sum_{i=1}^{d} \#\{n \in [0,2^{r+1} - 1] \mid S_d(r+1,n) = (\beta_i, \gamma_i)\} = 2 \times \#\{x \in \mathcal{M}_r \mid \chi_d(mx) = \alpha\}.$$

Now by definition of $B_d$

$$\sum_{i=1}^{d} \#\{n \in [0,2^{r+1} - 1] \mid S_d(r+1,n) = (\beta_i, \gamma_i)\} = \sum_{i=1}^{d} B_d(r+1, (\beta_i, \gamma_i)).$$

Thus,

$$\frac{\#\{x \in \mathcal{M}_r \mid \chi_d(mx) = \alpha\}}{2^r} - \frac{d}{d^2 - 1} = \sum_{i=1}^{d} \left( B_d(r+1, (\beta_i, \gamma_i)) \right) \cdot \frac{1}{2^{r+1}} - \frac{1}{d^2 - 1}.$$ 

Then we can use Theorem 1 and the triangular inequality to prove the first inequality of the theorem. Using this inequality for $\alpha \neq 0$ we prove the second inequality.

Taking $m$ as the all zeros 160 bits vector, this asymptotic result let us think that the values $\chi(c)$ for $c \in \mathcal{M}_{160}$ are well distributed modulo $d$ provided that $(F_{162}, d) = (F_{163}, d) = 1$.

## 5 A second family of EAC and an open problem

In order to generate a 160 bits integer, the author gives in [24] numerical results which show that the search for a chain whose length be less than 260 needs about $2^{23}$ tests using a heuristic from Montgomery [17]. With such chains, Algorithm 1 is much more performant than the classical SPA-resistant algorithms. We investigated the problem of choosing shorter chains in order to speed up the performances of algorithm 1. Once the length $\ell$ is fixed we have to deal with two constraints :

- the number $p$ of 1’s in the chain must be chosen so that the greatest integer generated is as near as $d \approx \frac{a}{2^{160}}$ as possible,
- because of the non-injectivity of $\chi$, $(\ell \choose p)$ must be greater than $2^{160}$ in order to hope that the integers generated reach most of the elements of $[1, 2^{160}]$. 

These two constraints lead us to study the set $\mathcal{M}_{\ell,p}$ where $p < \ell/2$.

**Theorem 3.** Let $(p, \ell) \in \mathbb{N}^2$ such that $0 < 2p < \ell$. Let $F_i$ be the $i$th Fibonacci, $\alpha_p = \frac{(1+\sqrt{5})^p + (1-\sqrt{5})^p}{2}$, $\beta_p = \frac{(1+\sqrt{5})^p - (1-\sqrt{5})^p}{2\sqrt{2}}$, then

i) For all $m \in M_{\ell,p}$ we have

$$F_{\ell-p+4} + pF_{\ell-p+2} \leq \chi(m) \leq F_{\ell-2p+4}(\alpha_p + \beta_p) + \beta_pF_{\ell-2p+2}.$$ 

ii) The lower bound is reached if and only if $m = 1^p0^{\ell-p}$ or $m = 0^{\ell-p}1^p$.

iii) The upper bound is reached if and only if $m = (01)^p0^{\ell-2p}$ or $m = 0^{\ell-2p}(10)^p$.

**Proof.** See annex.

To improve the performances of Algorithm 1, we choose to use chains whose length is 240. In this case $p = 80$ seems to be the best choice with respect to our two constraints. With such parameters, we can randomly generate about $2^{216}$ chains computing integers in the interval $[2^{117.7}, 2^{158.9}]$. Unfortunately, it seems to be a hard problem to compute the number of distinct integers generated in this way.

This naturally leads us to consider the set $\mathcal{M}_{3p,p}$. Numerical experiments for some values of $p$ let us think that the cardinality of $\chi(\mathcal{M}_{3p,p})$ is near from $2^{2p}$. Notice that from the preceding theorem, it can be proved that the upper bound for $\ell = 3p$ is equivalent to $\gamma\left(\frac{(1+\sqrt{5})^p + (1-\sqrt{5})^p}{2}\right)^p$ where $\gamma \in \mathbb{R}^+$ which is more or less equal to $2^{1.96p}$. We end this section with an open problem: What is the cardinality of $\chi(\mathcal{M}_{3p,p})$? The good performances of this method (see next section) make this problem of interest.

### 6 Comparison

In this section, we compare our Euclidean chain generation methods to several other Simple Power Analysis resistant methods. We focus our study on 160-bit and 224-bit integers and give comparisons with methods using a random or a fixed base point. We make the classical assumption that a field squaring (S) is equal to 0.8 time a field multiplication (M), except for unified formulae, that requires $M = S$. Tables 2 and 1 summarize our study.

In this paper we have proposed three different ways to compute a point on the curve from an Euclidean addition chain:

**Method 1:** use a chain from $\mathcal{M}_{160}^{0}$ for curves of order about $2^{224}$.

**Method 2:** use a chain from $\mathcal{M}_{160}^{0}$ for curves of order about $2^{160}$ (using the results on Stern sequences).

**Method 3:** use a chain from $\mathcal{M}_{240,80}$ for curves of order about $2^{160}$.

#### 6.1 Random base point

For curves of order $2^{224}$, using Algorithm 1, we only obtain with Method 1 $2^{160}$ different possible keys, which means that we propose a trade-off between security and efficiency. This is quite similar to the approaches using spare keys as an example. The cost of a scalar multiplication would be 2104M for the classical approach and 1784M if we compute only the $x$ coordinate.

For curves of order about $2^{160}$, we obtain that the cost of a scalar multiplication with Method 3 would be 1576M for the classical approach and 1336M if we compute only the $x$ coordinate. With Method 2, we know that the obtained integers are asymptotically well distributed modulo $d$, where $d$ is the order of the curve. The cost of a scalar multiplication is the same than the one obtained for curves of order $2^{224}$.

We compare these methods to various other SPA algorithmic countermeasure:
Table 1. Comparison of scalar multiplication methods (224-bit scalar)

- Dummy operations consist of adding a dummy point addition during the double-and-add algorithm, when the current bit is a 0.
- The Montgomery ladder is a SPA resistant algorithm from Peter Montgomery [25], performing one doubling and one addition for each bit of the scalar. It is really efficient only on Montgomery curve.
- Unified formulae allow to perform doubling and addition with the same formulae on specific curve shapes. They can be then combined with the NAF representation for scalar multiplication. The cost of the unified operation is 11M, 12M and 14M on Edwards [7], Hessian [26] and Jacobi curves [27] respectively.
- Möller proposed a modified version of Brauer (2\textsuperscript{w}-ary) algorithm [28]. Using precomputations, its pattern is independent from the scalar itself.

6.2 Fixed base point

In the context of a fixed base point we propose a new scalar multiplication scheme based on our chains generation method.

Notice that in Methods 1 and 2, the 160 first steps of Algorithm 1 are fixed, independently of the scalar, and correspond to big steps. Hence we can precompute and store the points \( F_{162}P \) and \( F_{163}P \) and then generate random chains of length 160. We compare our method to the classical comb method.

7 Conclusions

In this paper, we proposed two different methods to generate short chains, in order to use them in the ECC context. Table 1 and 2 show that our method provides good results in various situations. In particular, our new scheme in the context of a fixed base point is competitive against actual methods and much faster when using similar amount of storage (Methods 1 and 2). For random base point methods, we are even faster than Montgomery’s algorithm, when the computation of the \( y \)-coordinate is not required (Methods 1 and 3).
More important, we have given the first practical results on Euclidean addition chain random generation. We have proved essential results on the value, size, and distribution of such chains and the integers they compute, that will be central for any further investigation.

References

8 Annex (proof of theorem 3)

From now on, we will denote by $\overline{m}$, the value $\chi(m)$. Let us set $M = \sup \{ \overline{m} \mid m \in \mathcal{M}_{\ell,p} \}$ and $I = \inf \{ \overline{m} \mid m \in \mathcal{M}_{\ell,p} \}$. If $m \in \mathcal{M}_{\ell,p}$ is not one of the words of the points $ii)$ (resp. $iii)$, we will propose $m' \in \mathcal{M}_{\ell,p}$ such that $\overline{m'} < \overline{m}$ (resp. $\overline{m'} > \overline{m}$). The lemmas in the two following subsections give the details about the words $m'$ we use to compare.

We first look for $m \in \mathcal{M}_{\ell,p}$ such that $\overline{m} = I$. First suppose that two 1’s in the word $m$ are separated by one 0 or more. Then we can consider $(m,n,s) \in (\mathbb{N}_0)^3$ and $(a,b) \in \mathbb{N}^2$ and $(x,y) \in \mathcal{M}_a \times \mathcal{M}_b$ such that $m$ is one of the words

$$1^m0^n1^s, \quad x10^m1^s0y \quad \text{or} \quad y01^a0^n1x.$$

We won’t consider the third case because it is the symmetric of the second one. The lemma 2 shows that $1^m0^n1^s > 1^{m+1}0^n$ and that $x10^m1^s0y > x1^{n+1}0^m1^s+y$. So if $\overline{m} = I$, there are
no 0 between two 1 of the word \(m\), and so there are integers \(a\) and \(c\) such that \(m = 0^a10^c\). From lemma 3 we show that \(a = 0\) (and so \(c = \ell - p\)) or \(c = 0\) (and so \(a = \ell - p\)).

Now we look for \(m \in \mathcal{M}_{\ell,p}\) such that \(\overline{m} = M\). If there are two consecutive 1’s in the word \(m\), as \(2p < \ell\) the word \(m\) will also have two consecutive 0’s. We can consider the symmetry such that a subword 00 appears in \(m\) before a subword 11. In this case there exists \((a, b, n) \in \mathbb{N}^3\) and \((x, y) \in \mathcal{M}_a \times \mathcal{M}_b\) such that \(m = x00(10)^n11y\). In this case, we have from lemma 4 that \(\overline{m} < x(01)^{n+2}y\). Now assume that there are no subword 11 in \(m\). If two 1’s in the word \(m\) are separated by two 0’s or more, then there exists \((m, n, a, b) \in (\mathbb{N}^*)^2 \times \mathbb{N}^2\), \(x \in \mathcal{M}_a\) and \(y \in \mathcal{M}_b\) such that \(m\) is one of the words

\[
x010^n(01)^m, \quad x010^n(01)^m0, \quad x010^n(01)^m02y,
\]

\[
10^n(01)^m, \quad 10^n(01)^m0, \quad 10^n(01)^m2y. 
\]

The Lemmas (5) and (6) give us in any case \(m' \in \mathcal{M}_{\ell,p}\) such that \(\overline{m'} < \overline{m}\). We have just proved that 11 is not a subword of \(m\), and that two 1’s of \(m\) are separated by exactly one 0. So the word \(m\) or its symmetric is \(0^n(01)^m0^c\) where \((a, c) \in \mathbb{N} \times \mathbb{N}\). The lemma (7) shows that \(M\) is reached when \(a = 0\) (so \(c = \ell - 2p\)) or when \(c = 1\) (so \(a = \ell - 2p - 1\)). Then the point iii) is proved.

Now we just have to apply Proposition 1 to deduce \(i)\) from \(ii)\) and \(iii)\).

### 8.1 Lemmas to find the lower bound.

#### Lemma 2.

Let \((m, n, s) \in (\mathbb{N}^*)^3\) and \((a, b) \in \mathbb{N}^2\). Let \(x \in \mathcal{M}_a\) and \(y \in \mathcal{M}_b\). We have

\[
i) \quad \frac{1}{m}(0^n0^m1^s) > \frac{1}{m+n}(0^n0^{m+n}) \quad \text{and} \quad \\
ii) \quad \frac{1}{x10^n1^s0^{m+n}} > \frac{1}{x1^n0^{m+n}1^s}.
\]

**Proof.** For the point \(i)\), we compute

\[
\frac{1}{m}(0^n0^m1^s) = (1, 2) \begin{pmatrix} m \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{pmatrix} \begin{pmatrix} s \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \text{so}
\]

\[
\frac{1}{m+n}(0^n0^{m+n}) = (s + 1)F_{n-1} + ((s + 1)(m + 2) + 1)F_n + (m + 2)F_{n+1}.
\]

Also, \(\frac{1}{m+n}(0^n) = (1, 2) \begin{pmatrix} m + s \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \text{so}
\]

\[
\frac{1}{x1^n0^{m+n}} = (2 + m + s + 1)F_{n+1} + (2 + m + s)F_n.
\]

The difference between (2) and (3) is \(msF_n\), so it is positive in the conditions of the lemma. We can notice that there is equality when \(s = 0\) or when \(m = 0\), which can also be explained by the symmetry.

To prove the point \(ii)\), let us set \((v, u) = \psi(x)\). We have

\[
\psi(x10^m1^n0) = (v, u) \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} F_{m-1} & F_m \\ F_m & F_{m+1} \end{pmatrix} \begin{pmatrix} n \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}
\]

\[
= ((nF_m + F_{m+1})u + (nF_{m+1} + F_{m+2})v, F_{m+2}u + (F_{m+1} + (n + 1)F_{m+2})v + (u - v)nF_m).
\]

We also compute

\[
\psi(x1^n0^{m+1}) = (v, u) \begin{pmatrix} n + 1 \\ 0 \end{pmatrix} \begin{pmatrix} F_m & F_{m+1} \\ F_{m+1} & F_{m+2} \end{pmatrix}.
\]
Lemma 3. Let \((a, b, c) \in (\mathbb{N}^*)^3\), we have \(0^{n+1}0^c > 1^{4n+3}c\).

Proof. We first compute the left hand side

\[
0^{n+1}0^c = (1, 2) \begin{pmatrix} F_{a-1} & F_a \\ 0 & 1 \end{pmatrix} \begin{pmatrix} b \\ 0 \end{pmatrix} \begin{pmatrix} F_{c-1} & F_c \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}
\]

\[
= (F_{a+2}F_{c-1} + (bF_{a+2} + F_{a+3})F_c + F_{a+2}F_c + (bF_{a+2} + F_{a+3})F_{c+1} + 1)
\]

\[
= F_{a+2}F_{c+1} + F_{a+3}F_{c+2} + bF_{a+2}F_{c+2}.
\]

We also compute the right hand side

\[
1^{4n+2} = (1, 2) \begin{pmatrix} F_{a+c-1} & F_{a+c} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}
\]

\[
= (F_{a+c-1} + (b + 2)F_{a+c} + F_{a+c} + (b + 2)F_{a+c+1})
\]

\[
= F_{a+c+4} + bF_{a+c+2}.
\]

With eq. 1, page 4 we show that it is

\[
F_{a+2}F_{c+1} + F_{a+3}F_{c+2} + bF_{a+2}F_{c+2}.
\]

The difference between (6) and (7) is \(b(F_{a+c+2} - F_{a+2}F_{c+2}) = bF_cF_a\), which is positive in the conditions of the lemma. In the cases \(c = 0\) or \(a = 0\), there is equality which we already knew by the symmetry.

8.2 Lemmas to compute the upper bound.

Lemma 4. Let \((n, a, b) \in \mathbb{N}^3\). For all \(x \in \mathcal{M}_n\) and \(y \in \mathcal{M}_b\) we have

\[
x00(10)^a11y < x(01)^{a+1}y.
\]

Proof. We first compute

\[
M_0^2 (M_1M_0)^n M_1^2 = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} \alpha_n & 2\beta_n \\ \beta_n & \alpha_n \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \alpha_n + \beta_n & 3\alpha_n + 4\beta_n \\ \alpha_n + 2\beta_n & 4\alpha_n + 6\beta_n \end{pmatrix}.
\]

We set \((v, u) = \psi(x)\), so we have

\[
\psi(x00(10)^a11) = (v, u) \begin{pmatrix} \alpha_n + \beta_n & 3\alpha_n + 4\beta_n \\ \alpha_n + 2\beta_n & 4\alpha_n + 6\beta_n \end{pmatrix} = (v(\alpha_n + \beta_n) + u(\alpha_n + 2\beta_n), v(3\alpha_n + 4\beta_n) + u(4\alpha_n + 6\beta_n)).
\]

From the other hand

\[
M_0M_1 (M_0M_1)^n M_0M_1 = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} \alpha_n - \beta_n & \beta_n \\ \beta_n & \alpha_n + \beta_n \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}
\]

\[
= \begin{pmatrix} \alpha_n + \beta_n & 2\alpha_n + 3\beta_n \\ 2\alpha_n + 3\beta_n & 5\alpha_n + 7\beta_n \end{pmatrix},
\]

so

\[
\psi(x(01)^{a+2}) = (v(\alpha_n + \beta_n) + u(2\alpha_n + 3\beta_n), v(2\alpha_n + 3\beta_n) + u(5\alpha_n + 7\beta_n)).
\]

As \(v < u\) we can compare the vectors (8) and (9) components by components and then conclude.
Lemma 5. Let \((m,n,a,b)\in \mathbb{N}^2\times \mathbb{N}^2\). For all \(x \in \mathcal{M}_a\) and \(y \in \mathcal{M}_b\), we have

\begin{align*}
\text{i)} & \quad x_{101}^m(01)^m < x_{(01)}^{m+1}0^n \\
\text{ii)} & \quad x_{101}^m(01)^m0 < x_{(01)}^{m+1}0^n+1 \\
\text{iii)} & \quad x_{101}^m(01)^m0^n y < x_{(01)}^{m+1}0^n+1+y.
\end{align*}

Proof. We compute

\[
M_0M_1M_0^n = \begin{pmatrix}
0 & 1 \\
1 & 2
\end{pmatrix}
\begin{pmatrix}
F_{n-1} & F_n \\
F_n & F_{n+1}
\end{pmatrix}
= \begin{pmatrix}
F_n & F_{n+1} \\
F_{n+2} & F_{n+3}
\end{pmatrix}.
\]

We deduce

\[
M_0M_1M_0^n \ (M_0M_1)^m = \begin{pmatrix}
F_n & F_{n+1} \\
F_{n+2} & F_{n+3}
\end{pmatrix}
\begin{pmatrix}
\alpha_m - \beta_m & \beta_m \\
\beta_m & \alpha_m + \beta_m
\end{pmatrix}
\text{ and so}
\]

\[
M_0M_1M_0^n \ (M_0M_1)^m = \begin{pmatrix}
\alpha_mF_n + \beta_mF_{n-1} & \alpha_mF_n + \beta_mF_{n+1} \\
\alpha_mF_{n+2} + \beta_mF_{n+1} & \alpha_mF_{n+3} + \beta_mF_{n+2}
\end{pmatrix}.
\]

From the other hand, we can write \((M_0M_1)^{m+1}M_0^n = (M_0M_1)^m M_0M_1^n\) so

\[
(M_0M_1)^{m+1}M_0^n = \begin{pmatrix}
\alpha_m & \beta_m \\
\beta_m & \alpha_m + \beta_m
\end{pmatrix}
\begin{pmatrix}
F_n & F_{n+1} \\
F_{n+2} & F_{n+3}
\end{pmatrix},
\]

and then

\[
(M_0M_1)^{m+1}M_0^n = \begin{pmatrix}
\alpha_mF_n + \beta_mF_{n+1} & \alpha_mF_n + \beta_mF_{n+2} \\
\alpha_mF_{n+2} + \beta_mF_{n+1} & \alpha_mF_{n+3} + \beta_mF_{n+2}
\end{pmatrix}.
\]

Let us set \((v,u) = \psi(x)\).

\[
\psi(x_{101}^m(01)^m) = (v,u) \begin{pmatrix}
\alpha_mF_n + \beta_mF_{n-1} & \alpha_mF_n + \beta_mF_{n+2} \\
\alpha_mF_{n+2} + \beta_mF_{n+1} & \alpha_mF_{n+3} + \beta_mF_{n+2}
\end{pmatrix}
\begin{pmatrix}
1 \\
1
\end{pmatrix}
\]

\[
= v(\alpha_mF_{n+2} + \beta_mF_{n-1} + \beta_mF_{n+2}) + u(\alpha_mF_{n+4} + \beta_mF_{n+1} + \beta_mF_{n+4}).
\]

We also have

\[
\psi(x_{(01)}^{m+1}0^n) = (v,u) \begin{pmatrix}
\alpha_mF_n + \beta_mF_{n+1} & \alpha_mF_{n+1} + \beta_mF_{n+2} \\
\alpha_mF_{n+2} + \beta_mF_{n+1} + \beta_mF_{n+3} + \beta_mF_{n+4}
\end{pmatrix}
\begin{pmatrix}
1 \\
1
\end{pmatrix}
\]

\[
= v(\alpha_mF_{n+2} + \beta_mF_{n+1} + \beta_mF_{n+2}) + u(\alpha_mF_{n+4} + \beta_mF_{n+2} + \beta_mF_{n+4}).
\]

The difference between (13) and (12) is \(v\beta_mF_n + u\beta_mF_n\) so we deduce the first point.

To prove the point \text{ii)} we use (10) which gives

\[
M_0M_1M_0^n \ (M_0M_1)^m M_0^n = \begin{pmatrix}
\alpha_mF_{n+1} + \beta_mF_{n+2} & \alpha_mF_{n+2} + \beta_mF_{n-1} + \beta_mF_{n+2} \\
\alpha_mF_{n+3} + \beta_mF_{n+4} & \alpha_mF_{n+4} + \beta_mF_{n+1} + \beta_mF_{n+4}
\end{pmatrix}.
\]

Let us set \((v,u) = \psi(x)\). We have

\[
\overline{x_{101}^m(01)^m0} = v(\alpha_mF_{n+3} + \beta_mF_{n-1} + 2\beta_mF_{n+2})
\]

\[
+ u(\alpha_mF_{n+5} + \beta_mF_{n+1} + 2\beta_mF_{n+4}).
\]

From (11) we deduce

\[
(M_0M_2)^{m+1}M_0^n = \begin{pmatrix}
\alpha_mF_{n+1} + \beta_mF_{n+2} & \alpha_mF_{n+2} + \beta_mF_{n+3} \\
\alpha_mF_{n+3} + \beta_mF_{n+4} & \alpha_mF_{n+4} + \beta_mF_{n+2} + \beta_mF_{n+4}
\end{pmatrix}.
\]

So

\[
\overline{x_{(01)}^{m+1}0^n+1} = v(\alpha_mF_{n+3} + \beta_mF_{n+4}) + u(\alpha_mF_{n+5} + \beta_mF_{n+3} + \beta_mF_{n+5}).
\]
The difference between (17) and (15) is $v \beta_m F_n$ so we have the positivity. To prove iii) we compute from (14) and (16)
\[
M_0 M_1 M_0^0 (M_0 M_1)^m M_0^2 = \left( \begin{array}{c}
\alpha_m F_{n+2} + \beta_m F_{n+1} + \beta_m F_{n+2} \\
\alpha_m F_{n+4} + \beta_m F_{n+3} + \beta_m F_{n+4}
\end{array} \right)
\]
and $(M_0 M_1)^{m+1} M_0^{m+2} = \left( \begin{array}{c}
\alpha_m F_{n+2} + \beta_m F_{n+3} \\
\alpha_m F_{n+4} + \beta_m F_{n+3} + \beta_m F_{n+4}
\end{array} \right)$.

We compare components by components and then deduce iii)

**Lemma 6.** Let $(m, n, b) \in (\mathbb{N}^*)^2 \times \mathbb{N}$ and $y \in M_b$. We have

i) $10^n(01)^m < (01)^{m+1}0^{n-1}$,
ii) $10^n(01)^m0 < (01)^{m+1}0^n$,
iii) $10^n(01)^m0^y < (01)^{m+1}0^{n+1}$.

**Proof.** We will use the computations of the proof of lemma 5. Let us consider that $\psi(x) = (1, 1)$. In this case $\psi(x0)$ would be $(1, 2, 1)$. We have $x010^n(01)^m = 10^n(01)^m$. So with (12) we have
\[
10^n(01)^m = \alpha_m F_{n+2} + \beta_m F_{n+1} + \beta_m F_{n+2} + \alpha_m F_{n+4} + \beta_m F_{n+1} + \beta_m F_{n+4}.
\]
With (11) we have
\[
10^{n+1}(01)^{m-1} = \alpha_m F_{n+1} + \beta_m F_{n+1} + \beta_m F_{n+2} + 2\alpha_m F_{n+3} + 2\beta_m F_{n+3} + 2\beta_m F_{n+3},
\]
so $10^{n+1}(01)^{m-1} - 10^n(01)^m = \alpha_m F_{n+1} + \beta_m F_{n+2}$, and then we have the point i) . In the same way, we also have
\[
10^n(01)^m0 = \alpha_m F_{n+3} + \beta_m F_{n+4} + 2\alpha_m F_{n+5} + \beta_m F_{n+5} + 2\beta_m F_{n+4},
\]
and
\[
10^{n+1}(01)^{m-1} = \alpha_m F_{n+2} + \beta_m F_{n+3} + \beta_m F_{n+4} + 2\alpha_m F_{n+6} + \beta_m F_{n+6} + \beta_m F_{n+6},
\]
so $10^{n+1}(01)^{m-1} - 10^n(01)^m = F_n(\alpha_m + 2\beta_m)$ and then we deduce the point ii).

Now, $\psi(10^n(01)^m0^2) = (1, 1) M_0 M_1 M_0^0 (M_0 M_1)^m M_0^2$, and with (15) we find
\[
\psi(10^n(01)^m0^2) = (\alpha_m(F_{n+2} + F_{n+4}) + \beta_m(F_{n+1} + F_{n+5}),
\alpha_m(F_{n+3} + F_{n+5}) + \beta_m(F_{n+1} + F_{n+5})).
\]
With (11) we compute
\[
\psi((01)^{m+1}0^{n+1}) = (\alpha_m(F_{n+1} + 2F_{n+3}) + \beta_m(F_{n+1} + 3F_{n+3}),
\alpha_m(F_{n+2} + 2F_{n+4}) + \beta_m(F_{n+2} + 3F_{n+4})).
\]
The difference of the first component of (21) by the first component of (20) is $\alpha_m F_{n-2} + \beta_m F_{n-2}$, and the difference of the second components is $\alpha_m F_n + \beta_m(2F_{n+4} - F_{n+1} - F_{n+1})$ so the point iii) is proved.

**Lemma 7.** Let $(a, b, c) \in \mathbb{N} \times \mathbb{N}^* \times \mathbb{N}$. If $c \neq 1$ and $a \neq 0$ then $0^a(01)^0^c < (01)^{0^a+c}$.

**Proof.** We compute
\[
0^a(01)^0^c = (1, 2)
\]

\[
\begin{pmatrix}
F_{a-1} & F_a \\
F_a & F_{a+1}
\end{pmatrix}
\begin{pmatrix}
\alpha_b - \beta_b & \beta_b \\
\beta_b & \alpha_b + \beta_b
\end{pmatrix}
\begin{pmatrix}
F_{c-1} & F_c \\
F_c & F_{c+1}
\end{pmatrix}
\begin{pmatrix}
1 \\
1
\end{pmatrix}
\]

\[
\begin{pmatrix}
F_{a-1} & F_a \\
F_a & F_{a+1}
\end{pmatrix}
\begin{pmatrix}
\alpha_b - \beta_b & \beta_b \\
\beta_b & \alpha_b + \beta_b
\end{pmatrix}
\begin{pmatrix}
F_{c-1} & F_c \\
F_c & F_{c+1}
\end{pmatrix}
\begin{pmatrix}
1 \\
1
\end{pmatrix}
\]
\begin{equation}
\alpha_b (F_{a+2}F_{c+1} + F_{a+3}F_{c+2}) + \beta_b (F_{a+1}F_{c+1} + F_{a+4}F_{c+2}).
\end{equation}

On the other hand

\begin{equation}
(01)^{\alpha+c} = (1, 2) \begin{pmatrix} \alpha_b - \beta_b & \beta_b \\ \beta_b & \alpha_b + \beta_b \end{pmatrix} \begin{pmatrix} F_{a+c-1} \\ F_{a+c} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \alpha_b F_{a+c+4} + \beta_b (F_{a+c+2} + F_{a+c+4}).
\end{equation}

Using eq. 1, we prove that the difference between (23) by (22) is \( \beta_b (F_{c+1} - F_c) \). It is zero if and only if \( a = 0 \) or \( c = 1 \), and positive otherwise.