

UNIVERSITY OF OXFORD



**Expanding the use of  
quasi-subfield polynomials**

Candidate: Marie Euler

supervised by Dr Christophe Petit

A dissertation submitted for the degree of

*Master of Mathematics and Foundations of Computer Science*

Trinity 2019  
2nd September 2019

## Acknowledgements

I would like to thank Dr Petit for his encouragements and guidance as my supervisor. Weekly discussions helped me to generate new ideas and to distinguish the promising leads among them.

I also want to thank the Booking.com Women in Technology Scholarship which financial support enabled me to spend this wonderful year in Oxford.

Moreover, I would like to thank Gary McGuire and Daniela Mueller for their transparency towards my supervisor and me. Indeed, Theorem 1.2 is in some sense similar to their results of [12] not yet published, and which they shared with us in July 2019. However, these results were established independently at the same time.

Last but not least, I want to warmly thank my family and friends for their encouragements all along the summer.

## Abstract

The supposed hardness of the elliptic curve discrete logarithm problem is crucial for modern cryptographic protocols. In 2018, the article *Quasi-subfield polynomials and the elliptic curve discrete logarithm problem* [11] by Huang et al. highlighted the potential of a specific class of polynomials to solve this problem at lower cost.

Following different tracks that were mentioned in this article, we were able to prove new results: we have exhibited and proved five more families of quasi-subfield polynomials. They are based on additive groups and multiplicative groups. Nonetheless, none of the found families allows us to beat already known ECDLP algorithms. We explained this obstruction in the case of linearized polynomials by proving a new tight lower bound. Finally, we briefly discuss how other algebraic groups could be used in this context.

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>From quasi-subfield polynomials to an ECDLP algorithm</b>	<b>7</b>
2.1	Previous ECDLP algorithms . . . . .	7
2.2	The quasi-subfield approach . . . . .	8
2.3	Complexity of the quasi-subfield approach . . . . .	9
<b>3</b>	<b>Use of additive subgroups</b>	<b>13</b>
3.1	Linearized polynomials . . . . .	13
3.2	How to find linearised quasi-subfield polynomials . . . . .	15
3.3	Linearized quasi-subfield polynomials with $n$ Mersenne . . . . .	21
<b>4</b>	<b>Lower bounds on <math>\beta</math> for linearized quasi-subfield polynomials</b>	<b>23</b>
4.1	The result . . . . .	23
4.2	The informal proof . . . . .	25
4.3	The formal proof . . . . .	29
4.4	Comparison with other lower bounds . . . . .	31
4.5	Consequences of this theorem . . . . .	32
<b>5</b>	<b>Use of multiplicative groups</b>	<b>33</b>
5.1	Polynomials based on multiplicative groups . . . . .	33
5.2	New families of quasi-subfield polynomials . . . . .	33
5.3	Critics of this definition of quasi-subfield polynomial . . . . .	38
<b>6</b>	<b>Use of other algebraic groups</b>	<b>38</b>
6.1	Torus . . . . .	38
6.2	Elliptic curves . . . . .	39
<b>7</b>	<b>Conclusion</b>	<b>41</b>
	<b>References</b>	<b>42</b>
<b>A</b>	<b>Appendix</b>	<b>43</b>
A.1	Omitted proofs . . . . .	43
A.2	Systematic search of quasi-subfield polynomials . . . . .	45

## Symbols used

$p$	a prime number
$q$	a prime power
$n$	an integer
$n'$	an integer smaller than $n$
$\tilde{n}$	an integer dividing $n$
$k$	an integer such that $n = k \cdot \tilde{n}$ .
	We often choose $q = p^k$ so that $\mathbb{F}_{p^n} = \mathbb{F}_{q^{\tilde{n}}}$
$P$	a monic polynomial in $\mathbb{F}_{p^n}[X]$
$L$	a monic linearized polynomial in $\mathbb{F}_{p^n}[X]$
$\sigma$	An element of $Gal(\mathbb{F}_{q^{\tilde{n}}}/\mathbb{F}_q)$
$d$	The $\sigma$ -degree of $L$
$\lambda$	a polynomial in $\mathbb{F}_{p^n}[X]$ such that $P = X^{p^{n'}} - \lambda(X)$
$\ell$	$\log_p(\lambda)$
$\beta$	$\frac{\ell n}{n'^2}$
$m$	an integer
$\alpha$	$m \cdot \frac{n'}{n}$
$c_{algo}$	constant appearing in the estimation of the complexity of the ECDLP algorithm, currently estimated as 4.876
$\mathcal{L}_{p,n}$	the set of linearized quasi-subfield polynomials in $\mathbb{F}_{p^n}[X]$
■	a symbol used to mark unknown coefficients

# 1 Introduction

The hardness of the discrete logarithm problem (for a cyclic group  $G = \langle g \rangle$  and  $h$ , compute  $k$  such that  $h = g^k$ , also called DLP) is an essential component of modern cryptography. This is indeed fundamental in the Diffie Hellman key exchange protocol which enables us to exchange cryptographic keys through an insecure channel. When applied using elliptic curves, it is called the elliptic curve discrete logarithm problem (ECDLP).

In 2018, quasi-subfield polynomials were introduced in [11] by Huang, Kusters, Petit, Yeo and Yun, with the aim of obtaining a new and more efficient algorithm than what is currently known to solve the elliptic curve discrete logarithm problem. Indeed, they permit to solve more quickly a polynomial system which is central in this new approach of the problem. They are defined in the following way:

**Definition 1.1** (Quasi-subfield polynomial). Let  $p$  prime and  $n \geq n'$  two integers. Then,  $P = X^{p^{n'}} - \lambda(X) \in \mathbb{F}_{p^n}[X]$  is a quasi-subfield polynomial if and only if  $P$  completely splits over  $\mathbb{F}_{p^n}$  and  $\beta(P) := \frac{\log_p(\deg \lambda) \cdot n}{n'^2} \leq 1$ . In the following, we will note  $\ell := \log_p(\deg \lambda)$ . It may not be an integer.

This definition is motivated by the polynomial  $X^{p^{n'}} - X$  i.e. the polynomial whose roots exactly define the base field if  $n' = 1$ , or if  $n' > 1$  and  $n'|n$  define a subfield of  $\mathbb{F}_{p^n}$ . However [7] already treated the case of the base field and [9] treated the case  $n$  composite. Moreover, the most frequent cases in cryptography use  $p$  and  $n$  primes. Therefore, we wanted to mainly focus on the case when  $n$  is prime. These *subfield polynomials* can naturally be generalized by allowing more low-degree terms, leading to Definition 1.1 (hence the name *quasi-subfield polynomials*). In order to avoid dealing with subfield polynomials, we will only consider  $\deg \lambda > 1$ , and thus only deal with  $\beta(P) > 0$ . When the signification is clear, we will often write  $\beta$  instead of  $\beta(P)$ .

The idea to keep in mind in the following part is that the lower  $\beta$  is, the better the ECDLP algorithm is. If  $\beta$  is close to 1, this approach is slightly better than the most naive approach to solve the ECDLP but it does not improve the best existing algorithms. To beat the latter, we would need  $\beta$  close to 0.1 or less. The original paper gives only one family of quasi-subfield polynomials:  $P = X + X^{q^{p_0}} + \dots + X^{q^{p_a}}$ , where  $q$  and  $q'$  are powers of  $p$ ,  $n$  is such that  $p^n = q^{p_a+1}$ , and  $p_i = 1 + q' + \dots + q^i$ . Nonetheless, for this family,  $\beta$  is really close to 1. A natural question is therefore whether other families of quasi-subfield polynomials exist, and what is the minimal possible value for  $\beta$ .

**Methodology and results** In this work, we studied different candidates for quasi-subfield polynomials. We grouped them according to the group structure of their roots: additive groups, multiplicative groups and other algebraic groups were considered. We gave rules to deduce new linearized (a special case of polynomials based on additive groups) quasi-subfield polynomials from known ones and then group them by equivalence classes. This helped us to exhibit two new families of linearized quasi-subfield polynomials. One of our families uses a Mersenne number for  $n$  and contradicts with a conjecture made in [11].

We then established the following lower bound on  $\beta$  in the case of *linearized* polynomials.

**Theorem 1.2.** Let  $L = X^{p^{n'}} - (a_\ell X^{p^\ell} + a_{\ell-1} X^{p^{\ell-1}} + \dots + a_0 X)$  a linearized polynomial with  $\ell \geq 1$  and  $\forall i, a_i \in \mathbb{F}_{p^n}$ . If  $L$  completely splits over  $\mathbb{F}_{p^n}$  then  $\beta = \ell.n/n'^2 \geq 3/4$ .

This is a major obstruction to the existence of *linearized* quasi-subfield polynomials able to beat the best known ECDLP algorithms.

We used the computer algebra system SageMath[2] through this work to guide our intuition and conjecture interesting families of quasi-subfield polynomials. Associated with a result of McGuire and Sheekey [13] characterizing linearized polynomials that completely split over their field of definition, this helped us to conjecture two new families of linearized quasi-subfield polynomials. SageMath also helped us conjecture three new families of quasi-subfield polynomials based on multiplicative groups. The lowest  $\beta$  encountered for a quasi-subfield polynomial in this dissertation is  $\frac{4}{9} \log_2(3) \simeq 0.7$  (with a multiplicative polynomial).

**Outline.** In Section 2, we will describe more explicitly the contribution of quasi-subfield polynomials to the elliptic curve discrete logarithm problem algorithm. We will in particular highlight the relation between the value of  $\beta$  and the complexity of this new approach and thus explicit the constraints required for this to run faster than the already known algorithms. In Section 3, we will give three new families of linearized quasi-subfield polynomials. We will notice that their  $\beta$  is really close to 1 and explain this result by a lower bound on  $\beta$  for *linearized* polynomials in Section 4. In Section 5, we will explore quasi-subfield polynomials based on multiplicative groups and we will exhibit some of them. Finally, in Section 6, we will initiate a study based on other algebraic groups such as the torus or elliptic curves. We conclude the thesis in Section 7.

## 2 From quasi-subfield polynomials to an ECDLP algorithm

Let us now provide more information about the contribution of quasi-subfield polynomials to the solving of the elliptic curve discrete logarithm problem. Therefore, we will first do a benchmark of the existing ECDLP algorithms. Then, we will present the algorithm introduced in [11] which uses quasi-subfield polynomials to solve the ECDLP. We will later recall its complexity and give more insight about the link between the value of  $\beta$  and the complexity of the algorithm.

### 2.1 Previous ECDLP algorithms

Let us consider an ECDLP instance: Let  $\mathcal{E}$  be an elliptic curve on  $K = \mathbb{F}_{p^n}$ ,  $P$  a point on the curve  $\mathcal{E}$  and  $Q$  a point in  $\langle P \rangle$ , the group generated by  $P$ . We are looking for  $k$  such that  $Q = kP$ . How do we compute  $k$ ? At what cost? Throughout this document, we will compare the complexity of the algorithm of [11] solving the ECDLP to two targets:

- $O(p^n)$  which is approximately the size of  $\langle P \rangle$ . It is often called the complexity of the exhaustive search or of brute-force algorithms: it corresponds to compute all the elements of  $\langle P \rangle$  until finding  $Q$ .
- $O(p^{n/2})$  which is approximately  $\sqrt{|\langle P \rangle|}$ . It can be obtained using generic algorithms such that Baby-Step-Giant-Step or Pollard-Rho [18].

For more information about these algorithms and other tracks to solve the ECDLP, the reader can consult *Recent progress on the elliptic curve discrete logarithm problem* [8] by Galbraith and Gaudry. It is also worth noticing that the two targets introduced here are associated to algorithms which can solve the discrete logarithm problem in any group. Therefore, we can hope that the new algorithm, which uses the structure of the group, has a better complexity.

If we want to consider the case where  $n$  is composite, we write  $n = \tilde{n}k$  and consider  $q = p^k$  so that  $\mathbb{F}_{p^n} = \mathbb{F}_{q^{\tilde{n}}}$ . In that case, we need to compare our results to the best done for this kind of field: Gaudry [9] succeeded in 2009 to find an algorithm solving the elliptic curve discrete logarithm problem on  $\mathbb{F}_{q^{\tilde{n}}}$  in  $O(q^{2-2/\tilde{n}})$  where the  $O(\cdot)$  notation hides a constant increasing quickly with  $\tilde{n}$ . For  $\tilde{n} = 2$ , it leads to an algorithm in  $O(p^{(n/2)(2-1)}) = O(p^{n/2})$  comparable with the generic

algorithm. For  $\tilde{n} = 3$ , it leads to an algorithm in  $O(p^{(n/3)(2-2/3)}) = O(p^{4/9n})$  slightly better than generic algorithms.

Diem also proved that there exists a sequence of prime powers  $Q_i = q_i^{n_i}$  with  $n_i \simeq \sqrt{\log(q_i)}$  such that the ECDLP in  $\mathcal{E}(F_{Q_i})$  can be solved in subexponential time [7]. It works any elliptic curve over  $F_{Q_i}$  and uses an approach similar to the one introduced below but with subfield polynomials instead of quasi-subfield polynomials. This was one of the motivation of this new approach.

## 2.2 The quasi-subfield approach

Let us now introduce the algorithm which uses quasi-subfield polynomials to solve the elliptic curve discrete logarithm problem. let us consider an ECDLP instance: Let  $\mathcal{E}$  be an elliptic curve on  $K = \mathbb{F}_{p^n}$ ,  $P \in \mathcal{E}$  and  $Q \in \langle P \rangle$ . We are looking for  $k$  such that  $Q = kP$ .

Let  $R \in \mathbb{F}_{p^n}[X]$  be a quasi-subfield polynomial. We define  $V$  as the set of the roots of  $R$  and  $\mathcal{F} := \{(x, y) \in \mathcal{E} | x \in V\}$ . We are looking for more than  $|V|$  relations of the shape:  $a_j P + b_j Q = P_1 + \dots + P_m$  with the  $P_i$  in  $\mathcal{F}$ . Indeed, if we succeed to find these relations, some linear algebra will give the value of  $k$  such that  $Q = kP$ .

In order to find these relations, we rely on an idea introduced by Semaev in [16] which uses Semaev summation polynomials: for an elliptic curve  $\mathcal{E}$  defined on a field  $K$  we can define  $S_r \in K[X]$  such that

$$S_r(x_1, \dots, x_r) = 0 \Leftrightarrow \exists (x_1, y_1), \dots, (x_r, y_r) \in \mathcal{E}, (x_1, y_1) + \dots + (x_r, y_r) = 0$$

Moreover, we can choose  $a_j, b_j$  uniformly at random in  $\mathbb{F}_{p^n}$  and consider  $a_j P + b_j Q = (X_j, Y_j)$ . Then, computing  $P_1, \dots, P_m$  such that  $a_j P + b_j Q = P_1 + \dots + P_m = (x_1, y_1) + \dots + (x_m, y_m)$  with the  $x_i$  in  $V$  amounts in finding  $x_1, \dots, x_m \in V$  such that  $(X_j, Y_j) + (x_1, -y_1) + \dots + (x_m, -y_m) = 0$  (we here only explicit the case where we can use the reduced Weierstrass equation of the elliptic curve, and thus have  $-(x, y) = (x, -y)$ ). Therefore, it boils down to finding  $x_1, \dots, x_m \in V$  such that  $S_{m+1}(X_i, x_1, \dots, x_m) = 0$  and then finding the associated  $y_i$ .

In order to solve the polynomial equation  $S_{m+1}(X_i, x_1, \dots, x_m) = 0$ , let us introduce the following tools:

- Let  $\mathcal{M}$  be the set of monomials in  $K[x_1, \dots, x_m]$ . Let  $i$  be a positive integer.

For  $f = \sum_{M \in \mathcal{M}} a_M M \in K[x_1, \dots, x_m]$ , we define  $F^i(f) = \sum_{M \in \mathcal{M}} a_M^{p^i} M$ .



- Let

$$\begin{aligned}\phi &: K[x_1, \dots, x_m] \rightarrow K[x_1, \dots, x_m] \\ f(x_1, \dots, x_m) &\mapsto F^{n'}(f)(\lambda(x_1); \dots, \lambda(x_m))\end{aligned}$$

$$\text{Then } f^{p^{n'}} \equiv \phi(f) \pmod{(x_1^{p^{n'}} - \lambda(x_1), \dots, x_m^{p^{n'}} - \lambda(x_m))}$$

- Let  $S^{(0)}(x_1, \dots, x_m) = S_{m+1}(X_i, x_1, \dots, x_m)$  and for  $k \in \{1, \dots, m-1\}$ ,  
 $S^{(k)}(x_1, \dots, x_m) = \phi(S^{(k-1)}(x_1, \dots, x_m))$

Then the point decomposition problem is reduced to solving  $\mathcal{S} = \{S^{(k)}\}_{k=1}^{m-1}$  which is a sparse polynomial system of  $m$  equations and  $m$  variables. Repeating this step with different  $a_j, b_j$  until we know the decomposition of  $|V|$  different points of the curve. Then, as said before, it is possible to recover, through linear algebra, the value of  $k$ .

### 2.3 Complexity of the quasi-subfield approach

Let us now recall how the original article gave an estimation of the complexity of this algorithm.

We know that  $\mathcal{S} = \{S^{(k)}\}_{k=1}^{m-1}$  is a sparse polynomial system of  $m$  equations and  $m$  variables. Therefore, it can be solved efficiently using Rojas' sparse resultant algorithm [15] and a univariate polynomial root finding algorithm such as BTA [3]. According to [11] (Lemma 3.1) the cost of this step is  $\tilde{O}(m^{5.188}(3p^\ell)^{c_{algo}m^2})$ . Here we introduce the notation  $c_{algo}$  instead of the value 4.876 present in Lemma 3.1 since we think this numerical value may be suboptimal. Moreover, it succeeds only with probability  $\frac{|\mathcal{F}|^m/m!}{p^n}$  since  $(X_i, Y_i)$  is a random point on  $\mathcal{E}$  with  $|\mathcal{E}| \simeq p^n$  and the number of sum of  $m$  points in  $\mathcal{F}$  is approximately  $|\mathcal{F}|^m/m!$ . Also, heuristically, half of the values in  $V$  are the x-coordinates of exactly two points on the curve so  $|\mathcal{F}| \simeq |V| = p^{n'}$ . Furthermore, we are looking for  $p^{n'}$  relations of this type. Therefore the cost of the relation search phase is  $p^{n'} \frac{m! \cdot p^n}{p^{n' \cdot m}} \tilde{O}(m^{5.188}(3p^\ell)^{c_{algo}m^2})$

Once all the  $p^{n'}$  relations are gathered, each of them involves  $m$  points. Therefore, the system built from this relations is sparse. Thus, a sparse linear algebra algorithm can be used to finish the computation [19]. It costs approximately  $mp^{2n'}$ . This gives the complete cost of the algorithm:  $m!p^{n-n' \cdot m+n'} \tilde{O}(m^{5.188}(3p^\ell)^{c_{algo}m^2}) + mp^{2n'}$ . Rewriting it to make  $\beta$  appear, we get the following estimation of the complexity:

**Proposition 2.1** (Complexity of the new algorithm). Let us consider a  $\beta$ -quasi-subfield polynomial  $P = X^{p^{n'}} - \lambda(X)$  and  $\ell = \log_p(\deg \lambda)$ . If  $|\mathcal{F}| \simeq |\mathcal{V}| \simeq p^{n'}$ , the complexity of this algorithm is

$$\tilde{O} \left( m! p^{n \left( 1 + c_{algo} \beta \left( \frac{n'm}{n} \right)^2 - \frac{n'm}{n} \right) + n'} m^{5.188 \mathfrak{Z} c_{algo} m^2} \right) + mp^{2n'}$$

where  $c_{algo}$  is a constant involved in the cost of the resolution of the system  $\mathcal{S}$  currently majored by 4.876.

We will try to find  $m$  which minimises this complexity. In the following section we will consider  $\alpha > 0$  as an abbreviation for  $n'm/n$ . We will assume that  $m$  is fixed.

**Proposition 2.2** (Best choice of parameters). We can observe the following results in order to optimize the complexity:

- The minimal complexity is obtained for  $\alpha = \alpha_\beta$  with  $\alpha_\beta := \frac{1}{2c_{algo}\beta}$ . Then, the complexity becomes  $\tilde{O} \left( p^{\max(2\alpha_\beta/m, 1 - \alpha_\beta(1/2 - 1/m))n} \right)$ .
- In order to beat brute force algorithms, it is required to have  $m > \max(2\alpha_\beta, 2)$ . So we have interest not to choose a very small integer for  $m$ .
- If  $\alpha_\beta < 2$  and  $m \gg 1$ , then the complexity can be rewritten as  $\tilde{O} \left( p^{(1 - \alpha_\beta/2)n} \right)$
- Therefore, to beat generic algorithms, we need  $\alpha_\beta > 1$

We can remark that the condition  $\alpha_\beta < 2$  is not really restrictive. Indeed for all the quasi-subfield polynomials exhibited in this dissertation, we will have  $\alpha_\beta < \alpha_{0.5} < 1$ .

*Proof.* Let us now prove these four results. The complexity of the algorithm is bounded by  $\tilde{O}(m! p^{n(1 + c_{algo}\beta(n'm/n)^2 - n'm/n) + n'} m^{5.188 \mathfrak{Z} c_{algo} m^2}) + mp^{2n'}$  which with the  $\alpha$ -notation and the fact that  $m$  is considered as a fixed integer, can be rewritten as  $\tilde{O}(p^{n(c_{algo}\beta\alpha^2 - \alpha + 1) + \alpha n/m} + p^{2\alpha n/m})$

Since  $c_{algo}\beta\alpha^2 - \alpha + 1$  is minimum for  $\alpha = \frac{1}{2c_{algo}\beta} = \alpha_\beta$  (we recall that we only consider  $\beta > 0$ ) and then has minimal value  $c_{algo}\beta \frac{1}{(c_{algo}\beta)^2} - \frac{1}{2c_{algo}\beta} + 1 = 1 - \frac{1}{4c_{algo}\beta} = 1 - \frac{\alpha_\beta}{2}$ , we get that the complexity can be rewritten as

$$\tilde{O} \left( p^{\max(2\alpha_\beta/m, 1 - \alpha_\beta(1/2 - 1/m))n} \right)$$

In order to beat the brute force algorithms (which corresponds to a complexity of  $O(p^n)$ ), what we need is to have on one side  $2\alpha_\beta/m < 1$  which is true as soon as  $m > 2\alpha_\beta$ , and on the other side,  $1 - \alpha_\beta(1/2 - 1/m) < 1$  ie  $m > 2$ . Hence  $m > \max(2\alpha_\beta, 2)$ .

Moreover, one can notice that  $2\alpha_\beta/m \leq 1 - \alpha_\beta(1/2 - 1/m)$  if only if  $\alpha_\beta(1/m + 1/2) \leq 1$ . Therefore if  $m \gg 1$  then  $\alpha_\beta \leq 2$  implies  $2\alpha_\beta/m \leq 1 - \alpha_\beta((1/2 - 1/m))$ , so we can rewrite the complexity as  $\tilde{O}(p^{(1-\alpha_\beta/2)n})$ .

Generic algorithms have a complexity of  $O(p^{n/2})$ , therefore we need  $\alpha_\beta > 1$  to run faster than them.  $\square$

Let us now observe some values in order to get some insights on the results we can hope to have. We approximate  $c_{algo}$  by 4.876.

$\beta$	$1 - \alpha_\beta/2$	$\max(2\alpha_\beta, 2)$	$\alpha_\beta$
1.0	0.949	2	0.103
0.8	0.936	2	0.128
0.6	0.915	2	0.171
0.4	0.872	2	0.256
0.2	0.744	2	0.513
0.15	0.658	2	0.684
0.1	0.487	2.05	1.025

Table of the relations between  $\beta$  and the complexity

We observe that for  $\beta = 1$ ,  $1 - \alpha_\beta/2 \simeq 0.95$  so we get a complexity slightly better than the one of brute force algorithms. This motivated the choice of the bound  $\beta \leq 1$  for the definition of quasi-subfield polynomials.

Also, we observe again here the fact that we beat generic algorithms for  $\alpha_\beta > 1$ , which for this specific value of  $c_{algo}$  implies  $\beta < 0.103$ .

Of course, the results of Proposition 2.2 uses the fact that we can choose any value for  $\alpha = n'm/n$ . However this would be an ideal case. Indeed this implies  $m = \alpha n/n'$ , which is rarely an integer. Thus it is unlikely that  $\alpha_\beta n/n'$  is an integer. In that sense, the results of Proposition 2.2 gives only lower bounds on the complexity. Therefore, we can ask to what extent can we change  $\alpha$  so that  $\alpha n/n'$  is an integer and the complexity is still interesting.

**Lemma 2.3** (Choice of  $\alpha$  with  $m$  integer ). Here are a few results about the realistic requirements to beat the other algorithms:

- If  $\alpha_\beta = \frac{1}{2 \cdot c_{algo}\beta} \geq \frac{9n'}{4n}$ , then it is possible to beat the exhaustive search.
- If  $\alpha_\beta > \frac{1 + \sqrt{1 + n'^2/n^2}}{2}$  and  $\frac{n'}{n} \ll \alpha_\beta - \sqrt{\alpha_\beta^2 - \alpha_\beta}$ , then it is possible to reach a complexity better than generic algorithms.

The proof can be found in appendix.

In the following, we will only keep the least restrictive condition which give us a good overview of when the polynomials would allow us to have an algorithm better than the exhaustive search. Therefore, we would like to suggest the following variant of quasi-subfield polynomials:

**Definition 2.4** (Full quasi-subfield polynomials). Let  $p$  prime and  $n \geq n'$  two integers. Then,  $P = X^{p^{n'}} - \lambda(X) \in \mathbb{F}_{p^n}[X]$  is a full quasi-subfield polynomial if and only if it matches the following three requirements :

- $P$  completely splits over  $\mathbb{F}_{p^n}$
- $\beta(P) = \frac{\log_p(\deg \lambda) \cdot n}{n'^2} \leq 1$
- $\alpha_\beta = \frac{1}{2 \cdot c_{algo}\beta} \geq \frac{9n'}{4n}$ .

Let us apply this definition to the quasi-subfield polynomials exhibited in the original article. There, it is shown that  $P_a = X + \sum_{i=0}^a X^{q^{p_i}}$ , (where  $q = p^k$ ,  $q' = p^r$ ,  $p_i = \sum_{j=0}^i q'^j$ ) is a 1-quasi-subfield polynomial in  $\mathbb{F}_{q^{p_{a+1}}}$ . The associated values are  $n = kp_{a+1}$ ,  $n' = kp_a$ ,  $\ell = kp_{a-1}$ . Moreover  $n/n' = kp_{a+1}/(kp_a) = (q'p_a + 1)/p_a \simeq q'$  so  $P_a$  is a full quasi-subfield polynomial if and only if  $q' > 21$  (for  $c_{algo} \simeq 4.876$ ).

$$\begin{aligned}
\text{Furthermore, } \beta &= \frac{p_{a+1} \cdot p_a - 1}{p_a^2} = \frac{(q' \cdot p_a + 1)(p_a - 1)/q'}{p_a^2} \\
&= \frac{p_a^2 - p_a + (p_a - 1)/q'}{p_a^2} \\
&= 1 - \frac{1}{p_a} \left( 1 - \frac{1}{q'} + \frac{1}{q' \cdot p_a} \right) \\
&\geq 1 - \frac{1}{2} \left( 1 + \frac{1}{2} \right) = 1/4 \quad (q' \geq 1 \text{ and } p_a \geq p_2 = 1 + q' \geq 2)
\end{aligned}$$

Thus  $\alpha_\beta < 1$ , therefore there is no hope of beating generic algorithms with this family. This justifies our need to find other families of quasi-subfield polynomials.

Now that we have motivated the interest of quasi-subfield polynomials to solve the ECDLP, and given more insights of what we expect from these polynomials, let us search some other families of quasi-subfield polynomials.

### 3 Use of additive subgroups

The previous family of quasi-subfield polynomials belong to a larger family called the linearized polynomials. These polynomials have the particularity that the group of their roots is additive. Since the only example of quasi-subfield polynomials we had was of this type, we dedicated a good part of our time to the search of similar quasi-subfield polynomials.

Let us now present the track we followed to find them. We will first introduce the linearized polynomials and some of their properties. Then, we will introduce how we found new families of quasi-subfield polynomials and how we grouped them in order to avoid repetition of similar polynomials. Finally, we will comment on the presence of Mersenne prime  $n$  in the exhibited families.

#### 3.1 Linearized polynomials

Let us begin by introducing linearized polynomials. Since we focus on the roots of these polynomials, we can admit without loss of generality that they are monic. Let us consider  $q$  a prime power:  $q = p^k$  with  $p$  prime and  $k \geq 1$ . Then, we now write  $n = \tilde{n}k$  so that  $\mathbb{F}_{p^n} = \mathbb{F}_{q^{\tilde{n}}}$ .

**Definition 3.1** (Linearized polynomials). Let  $f = a_0 + a_1X + \dots + X^d \in \mathbb{F}_{p^n}[X]$ .

Then  $L_{f,\sigma} = a_0X + a_1X^\sigma + \dots + X^{\sigma^d} \in \mathbb{F}_{p^n}[X]$  with  $\sigma \in \text{Gal}(\mathbb{F}_{p^n})$  is the linearized polynomial associated with  $f$  and  $\sigma$ . The  $\sigma$ -degree of  $L_{f,\sigma}$  is  $d$ .  $\sigma \in \text{Gal}(\mathbb{F}_{p^n})$  so  $X^\sigma$  is a notation for  $\sigma(X) = X^{q^s}$  with  $\gcd(s, \tilde{n}) = 1$ .

It is worth noticing that when we are considering the case  $k > 1$ , we are working with extension field with the degree of the extension being composite. The complexity of the best known algorithms for ECDLP in these fields is recalled in Section 2.

Gary McGuire and John Sheekey give in *A characterisation of the number of roots of linearized and projective polynomials in the field of coefficients* [13] a way to identify linearized polynomial which completely splits and this could be quasi-subfield polynomial. It uses the following matrices.

**Definition 3.2** (Companion matrix  $C_L$ ,  $A_L$ ). Let  $L = a_0X + a_1X^\sigma + \dots + X^{\sigma^d}$  a linearized polynomial with coefficients in  $\mathbb{F}_{p^n}$ . We can associate to  $L$  the matrix

$$C_L := \begin{bmatrix} 0 & 0 & \dots & 0 & -a_0 \\ 1 & 0 & \dots & 0 & -a_1 \\ 0 & 1 & \dots & 0 & -a_2 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & 1 & -a_{d-1} \end{bmatrix}.$$

Then, we also define the matrix  $A_L := C_L \cdot C_L^\sigma \cdot C_L^{\sigma^2} \dots C_L^{\sigma^{\tilde{n}-1}}$  with  $C_L^\sigma$  being the application of  $x \mapsto x^\sigma = x^{q^s}$  coefficient-wise. Note that  $A_L$  and  $C_L$  are square matrices of dimension  $d$ .

Then, the characterisation of completely splitting linearized polynomials relies on this useful result:

**Proposition 3.3** (Number of roots). Let  $L = a_0X + a_1X^\sigma + \dots + X^{\sigma^d}$  a linearized polynomial with coefficients in  $\mathbb{F}_{p^n}$ . The number of roots of  $L$  in  $\mathbb{F}_{q^{\tilde{n}}}$  is given by  $q^{n_1}$ , where  $n_1$  is the dimension of the eigenspace of  $A_L$  with eigenvalue 1. Hence,  $L$  completely splits over  $\mathbb{F}_{p^n}[X]$  if and only if  $A_L = I$ .

The maximum number of roots of  $L$  in  $\mathbb{F}_{q^{\tilde{n}}}$  is  $q^d$ . So  $L$  completely splits over  $\mathbb{F}_{q^{\tilde{n}}}$  only if  $s = 1$ . Therefore we will only consider the case  $s = 1$  in the following discussion. Hence, we can now replace  $\sigma$  by  $q$  in the expression of  $L$ . Moreover, one can observe any polynomial written as  $L_{f,p^k}$  can also be written as  $L_{g,p}$  with  $g = f(X^k)$ . Hence we will now always admit that  $q = p$  and write  $L_f$  as a shortcut for  $L_{f,p}$ .

Another property about completely splitting linearized polynomials is the following. It will be really useful when it comes to verifying that a linearized polynomial completely splits.

**Proposition 3.4** (Completely splitting). Let  $f = a_0 + a_1X + \dots + X^d \in \mathbb{F}_{p^n}[X]$ . Then the following properties are equivalent.

1.  $L_f(X)$  completely splits over  $\mathbb{F}_{p^n}[X]$
2.  $L_f(X)$  divides  $X^{p^n} - X$
3.  $f$  divides  $X^n - 1$

## 3.2 How to find linearised quasi-subfield polynomials

### 3.2.1 Rules to deduce new quasi-subfield polynomials from known ones

In order to simplify our search of quasi-subfield polynomials, we will first focus in ways to deduce new quasi-subfield polynomials from known ones. We will introduce two types of transformations: the first one will change the value of  $\beta$  and thus potentially improve it, but the value of the coefficients are hard to compute explicitly in the general case. The second one will keep the same  $\beta$ , thus will not produce more interesting linearised quasi-subfield polynomials but will allow us to group them by equivalence classes. Let us write  $\mathcal{L}_{p,n}$  for the set of linearized quasi-subfield polynomials in  $\mathbb{F}_{p^n}[X]$ .

The first way to obtain a linearized quasi-subfield polynomial from another one is what we will call in the following the inversion process.

**Proposition 3.5** (Inversion). Let  $f = a_0 + \dots + a_\ell X^\ell + X^{n'} \in \mathbb{F}_{p^n}[X]$  such that  $L_f \in \mathcal{L}_{p,n}$  and  $n' < n$ . Let  $g = (X^n - 1)/f$ . Then  $L_g \in \mathbb{F}_{p^n}[X]$  is a quasi-subfield polynomial. We say that  $L_g$  is the inverse of  $L_f$ .

*Proof.* By Proposition 3.4, we know that  $f|X^n - 1$  so  $g$  is well-defined.

Let  $g = b_0 + \dots + b_r X^r + X^{n-n'}$ . We will prove that  $r + n' \leq n - n' + \ell$ . Indeed,  $f.g = (X^{n'} + a_\ell X^\ell + \dots + a_0)(X^{n-n'} + b_r X^r + \dots + b_0) = X^n + b_r X^{r+n'} + a_\ell X^{\ell+n-n'} + \dots = X^n - 1$ . So, if  $r + n' > \ell + n - n'$  then the coefficient of  $X^{r+n'}$  in  $f.g$  comes exclusively from  $(b_r \cdot X^r) \cdot X^{n'}$  and thus is not zero. Therefore,

$$\begin{aligned} \beta(L_g) &= \frac{n.r}{(n-n')^2} \leq \frac{n.(n-2n'+\ell)}{(n-n')^2} = 1 - \frac{n'^2 - \ell.n}{(n-n')^2} \\ &\leq 1 - \left(\frac{n'}{n-n'}\right)^2 (1 - \beta(L_f)) \leq 1 \end{aligned}$$

□

Let us now introduce transformations that keep the value of  $\beta$  unchanged.

**Proposition 3.6** (Transformations keeping same  $\beta$ ). Let  $k \geq 1$ ,  $\alpha \in \mathbb{F}_{p^n}$  with  $\alpha^n = 1$ ,  $\gamma \in \mathbb{F}_{p^n}^*$ . Let  $f = a_0 + \dots + a_\ell X^\ell + X^{n'} \in \mathbb{F}_{p^n}[X]$ .

Then the following propositions are equivalent:

- (1)  $L_f \in \mathcal{L}_{p,n}$
- (2)  $L_{f,p^k} = L_{f(X^k)} = a_0 X + a_1 X^{p^k} + \dots + a_\ell X^{p^{k.\ell}} + X^{p^{k.n'}} \in \mathcal{L}_{p,kn}$
- (3)  $\alpha^{-n'} L_{f(\alpha.X)} = \alpha^{-n'} (a_0 X + a_1 \alpha.X^p + \dots + a_\ell \alpha^\ell.X^{p^\ell} + \alpha^{n'}.X^{p^{n'}}) \in \mathcal{L}_{p,n}$
- (4)  $\gamma^{-p^{n'}} L_f(\gamma.X) = \gamma^{-p^{n'}} (a_0 X + a_1 (\gamma.X)^p + \dots + a_\ell (\gamma.X)^{p^\ell} + (\gamma.X)^{p^{n'}}) \in \mathcal{L}_{p,n}$

*Proof.* One can observe that these four quasi-subfield polynomials have the same  $\beta$ . Therefore, we only have to show that the conditions for splitting are equivalent.

The equivalence between (1) and (2) comes directly from Proposition 3.4. Indeed,

$$\begin{aligned} (1) \ L_f \in \mathcal{L}_{p,n} \Leftrightarrow f|X^n - 1 &\Leftrightarrow f(X^k)|X^{kn} - 1 \text{ in } \mathbb{F}_{p^n}[X] \\ &\Leftrightarrow L_{f(X^k)} \in \mathcal{L}_{p,kn} \text{ (3) (we recall that the} \\ &\quad a_i \text{ are fixed in } \mathbb{F}_{p^n}) \end{aligned}$$

Let us now prove that (1) and (3) are equivalent. Indeed,

$$\begin{aligned} (1) \ L_f \in \mathcal{L}_{p,n} \Leftrightarrow f|X^n - 1 &\Leftrightarrow f(\alpha.X)|(\alpha.X)^n - 1 \\ &\Leftrightarrow f(\alpha.X)|X^n - 1 \text{ since } \alpha^n = 1 \\ &\Leftrightarrow \alpha^{-n'} f(\alpha.X)|X^n - 1 \\ &\Leftrightarrow \alpha^{-n'} L_{f(\alpha.X)} \in \mathcal{L}_{p,n} \text{ (3)} \end{aligned}$$

Finally, replacing  $X$  by  $\gamma.X$  clearly does not change the fact that the polynomial split, therefore (1)  $\Leftrightarrow$  (4) is trivial.  $\square$

One can also observe that  $n/n'$  is not changed by any of these transformations. So we would also write all these equivalences with full linearized quasi-subfield polynomials.

Hence since none of these transformations changes the value of  $\beta$  and they all send (full) linearized quasi-subfield polynomials on other (full) linearized quasi-subfield polynomials, we can define equivalence classes by saying that two linearized quasi-subfield polynomials are equivalent to each other if one can be obtained from the other with one of the previous transformations. Obviously, since the transformations do not change the value of  $\beta$ , we are only interested by finding one representative of each class.

Let us now consider polynomials with only coefficients in the base field  $\mathbb{F}_p$ . Then if  $L$  is of the shape  $a_0X + a_1X^{p^k} + \dots + a_{n'}X^{p^{kn'}} \in \mathcal{L}_{p,kn}$ , it is equivalent to  $a_0X + a_1X^{p^k} + \dots + a_{n'}X^{p^{kn'}} \in \mathcal{L}_{p,n}$ . Therefore, we may reduce the search of representative of each class to polynomial of the shape  $L = a_0X + a_1X^{p^k} + \dots + a_\ell X^{p^\ell} + X^{p^{n'}} \in \mathbb{F}_{p^n}[X]$  with  $\{i \geq 1, a_i \neq 0\} \cup \{n\}$  coprime setwise.

One can also notice that transformation (1)  $\Leftrightarrow$  (3) cannot often be used. Indeed if  $n$  is prime then  $\alpha^n = 1$  implies  $n|p^n - 1$ . Moreover the transformation (1)  $\Leftrightarrow$  (4) changes the values of the coefficients only when considering  $\gamma \in \mathbb{F}_{p^n}$  *mathbbF*<sub>p</sub>, therefore it is not useful when we only consider polynomials with coefficients in  $\mathbb{F}_p$ .



### 3.2.2 New families of quasi-subfield polynomials

Let us now look at how we could do a systematic search of representatives of each class of equivalence of quasi-subfield polynomials.

As seen before, we will search for linearized polynomials  $L = a_0X + a_1X^p + \dots + a_\ell X^{p^\ell} + X^{p^{n'}}$  in  $\mathbb{F}[X]$  which verify  $\beta \leq 1$ , completely split over  $\mathbb{F}_{p^n}$  and such that  $\{i \geq 1, a_i \neq 0\} \cup \{n\}$  are coprime setwise.

We recall that we restrict the search to polynomials in  $\mathbb{F}_p[X]$ . Therefore  $C_L \in M_d(\mathbb{F}_p)$ , thus  $A_L = C_L^n$ . Hence, Proposition 3.3 says that  $L$  splits over  $\mathbb{F}_{p^n}$  if and only if  $C_L^n = I$ . A first idea could then be to consider a fixed  $n$ , compute  $C_{L_f}^n$  for all the linearized polynomial  $L_f \in \mathbb{F}_p[X]$  of degree less than  $n$  and check whether it is the identity. When it is the identity, it means that  $L_f$  splits in  $\mathbb{F}_{p^n}$ . After that, we still have to verify that  $\beta(L_f) < 1$ . However, in this plan, as  $n$  grows, we have to consider more and more  $L_f$  and the chance to find a quasi-subfield polynomial collapses.

Another idea is to consider a fixed linearized polynomial  $L$  and search for the smaller  $n$  such that  $L$  splits over  $\mathbb{F}_{p^n}$ . This amounts to searching for  $n$  such that  $C_L^n = I$ : finding the order of  $C_L$ . Note that  $C_L$  is in  $GL_{n'}(\mathbb{F}_p)$  since  $\det C_L = (-1)^{n'} a_0 \neq 0$  (if  $a_0 = 0$  then  $0$  is a root of  $L$  with multiplicity at least  $p$  so  $L$  does not split completely). Hence  $C_L$  belongs to a finite group. Therefore  $n$  exists. Moreover as we also want  $\beta(L) = n \cdot \ell / (n')^2 \leq 1$ , we only have to check the  $C_L^k$  with  $k < n'^2 / \ell$ . If we find such a  $k$ , then  $L_f \in \mathcal{L}_{p,k}$ . In order to accelerate the computation, we can verify before starting the computation of the  $C_L^k$ , that  $L$  is not in the equivalence classes of the known quasi-subfield polynomials.

Therefore we can write an algorithm (presented extensively in Appendix B) to produce a set of representatives of the previously defined quasi-subfield polynomials. It outputs results of this kind.

$f$	$n$	$\beta$	$p$
$X^2 + X + 1$	3	0.75	2,3,5,7
$X^3 + X + 1$	7	0.77	2
$X^3 + X + 1$	8	0.88	3
$X^3 + X^2 + X + 1$	4	0.88	2,3,5,7
$X^4 + X + 1$	15	0.93	2
$X^4 + X + 1$	13	0.81	3
$X^4 + X^2 + X + 1$	7	0.87	2
$X^4 + X^3 + X^2 + X + 1$	5	0.93	2
$X^5 - X^3 + X^2 + X + 1$	8	0.96	3
	$\vdots$		

Outputs of the algorithm (In order to improve the readability we list the value of  $f$  instead of the quasi-subfield polynomials  $L_f$ .)

Observing patterns in them allowed us to conjecture new types of quasi-subfield polynomials. Of course, here we only present one representative per equivalence class. Therefore other quasi-subfield polynomials can be obtained by using the rules listed in Proposition 3.6. Moreover, let us recall that this list does not cover all the equivalence classes. It was only conjectured from what was found with small  $n$  and small  $p$ .

**Proposition 3.7** (Families of linearized quasi-subfield polynomials). The following types of linearized polynomials are quasi-subfield polynomials:

**Type 1**  $L_h$  with  $h = 1 + X^{p_0} + \dots + X^{p_a}$ , where  $q = p^r$ ,  $r \geq 0$ ,  $n = p_{d+1}$ ,  $p_i = 1 + q + \dots + q^i$  and  $a \geq 2$   $\beta = 1 - \frac{1}{p_a}(1 - \frac{p_a-1}{q^{p_a}})$ . It is the family introduced in [11]

**Type 1bis**  $X + X^p + X^{p^2} + \dots + X^{p^{n-1}}$ ,  $n' = n - 1$ ,  $\beta = 1 - \frac{1}{(n-1)^2}$

**Type 2**  $L_{f_a}$  with  $f_a = \begin{cases} 1 + X^{q^{-1}} + \dots + X^{q^{d-1}} & \text{if } a = 0 \\ a + X + X^q + \dots + X^{q^d} & \text{otherwise} \end{cases}$ ,  $n = q^{d+1} - 1$ ,  
 $q = p^r$ ,  $r \geq 1$ ,  $a \in \mathbb{F}_q$ ,  $\beta = 1 - \frac{q^{d-1}}{(1+q+\dots+q^{d-1})^2}$

**Type 3** Inverses of type 1 and inverses of type 2.

We will now prove these families which were initially introduced as conjectures.

*Proof.* The type 1 is proven in the original paper [11].

Moreover it is obvious that  $L_{X^{-1}}$  is a quasi-subfield in  $\mathbb{F}_{p^n}$  for any  $p$  prime and  $n$  (tolerating here  $\ell = 0$ ). Therefore its inverse (see Proposition 3.5)  $L_{(X^{n-1})/(X^{-1})} = L_{1+X+X^2+\dots+X^{n-1}}$  is a quasi-subfield in  $\mathbb{F}_p^n$ . This proves the type 1bis. One can notice that it is in fact a particular case of Type 1 (when  $r = 0$ ).

Looking at type 2, we need to compute the factorisation of  $X^{q^{d+1}-1} - 1$ . It may be easier to compute this by looking at  $X(X^{q^{d+1}-1} - 1) = X^{p^{r(d+1)}} - X$  since the Frobenius is easy to compute in  $\mathbb{F}_{p^n}$ . One can observe for any  $a \in \mathbb{F}_q$ ,

$$\begin{aligned} (X^{q^d} + \dots + X + a)^q - (X^{q^d} + \dots + X + a) &= X^{q^{d+1}} + X^{q^d} + \dots + X^q + a^q \\ &\quad - X^{q^d} - \dots - X^q - X - a \\ &= X^{q^{d+1}} - X \end{aligned}$$

Indeed since  $a \in \mathbb{F}_q$ , we have that  $a^q = a$ . Hence writing  $g_a = a + X + X^q + X^{q^2} + \dots + X^{q^d}$ , we have that  $X^{q^{d+1}} - X = g_a^q - g_a$ , hence  $g_a | X^{q^{d+1}} - X$ . Let  $a \neq b$ , then  $\gcd(g_a, g_b) = \gcd(g_a, g_a - g_b) = \gcd(g_a, a - b) = a - b$  where the greatest common divisor is defined up to multiplication by an invertible constant. Thus  $g_a$  and  $g_b$  are coprime. Thus  $\prod_{a \in \mathbb{F}_p} g_a | X^{q^{d+1}} - X$ . But  $\deg(\prod_{a \in \mathbb{F}_q} g_a) = q \cdot q^d = \deg(X^{q^{d+1}} - X)$ . Therefore  $\prod_{a \in \mathbb{F}_q} g_a = X^{q^{d+1}} - X$ . Rewriting it, we get,  $(X^{q^d} + \dots + X) \prod_{a \in \mathbb{F}_q^*} g_a = X^{q^{d+1}} - X$ . Thus,  $(X^{q^d-1} + \dots + 1) \prod_{a \in \mathbb{F}_q^*} g_a = X^{p^{d+1}-1} - 1 = X^n - 1$ . But if  $a \neq 0$ ,  $g_a = f_a$  and  $f_0 = X^{q^d-1} + \dots + 1$ . That is why,  $\prod_{a \in \mathbb{F}_q} f_a = X^n - 1$ . This gives that polynomials of the type 2 completely split over  $\mathbb{F}_{p^n}$ . Moreover, we also have to verify that  $\beta \leq 1$ .

But if  $a = 0$ , then  $n' = q^d - 1$  and  $\ell = q^{d-1} - 1$  so,

$$\begin{aligned} \beta &= \frac{(q^{d-1}-1)(q^{d+1}-1)}{(q^d-1)^2} = 1 - \frac{q^{d+1}+q^{d-1}-2q^d}{(q^d-1)^2} = 1 - \frac{q^{d-1}(q-1)^2}{(q^d-1)^2} \\ &= 1 - \frac{q^{d-1}}{(1+q+\dots+q^{d-1})^2} < 1 \end{aligned}$$

If  $a \neq 0$ , then  $n = q^{d+1} - 1$ ,  $n' = q^d$ ,  $\ell = q^{d-1}$ , so  $\beta = \frac{q^{d-1}(q^{d+1}-1)}{q^{2d}} = \frac{q^{d+1}-1}{q^{d+1}} = 1 - \frac{1}{q^{d+1}} < 1$

In any case  $n'/n \simeq 1/q$  and  $\alpha_\beta \geq 1/(2c_{algo}) \simeq 1/10$ , thus  $L_{f_a}$  is a full quasi-subfield polynomial as soon as  $q \geq 22$ .

Regarding Type 3, by Proposition 3.5 there is nothing to prove anymore. About the exact values, we know that for Type 1 we have,  $Xh^q = h + X^n - 1$  thus  $X^n - 1 = h(Xh^{q-1} - 1)$  and the inverse of  $L_h$  is  $L_{Xh^{q-1}-1}$ . For Type 2, we have  $\prod_{a \in \mathbb{F}_p} f_a = X^n - 1$ , thus the inverse of  $L_{f_a}$  is  $L_{\prod_{b \neq a} f_b}$ .  $\square$

We will now try to classify the results output by our algorithm when asking for representative of the equivalence classes for  $p$  being 2,3,5 or 7 and  $n'$  equal or

less than 16. We mark by a cross when the linearized polynomial belongs to the category, except for the last category where we give the value of the inverse.

$f$	$n$	$\beta$	$p$	T1	T2	T3
$X^2 + X + 1$	3	0.7	2	X	X	
$X^2 + X + 1$	3	0.7	3,5,7	X		
$X^3 + X + 1$	7	0.7	2	X	X	
$X^3 + X + 1$	8	0.8	3		X	
$X^3 + X^2 + X + 1$	4	0.8	2,3,5,7	X		
$X^4 + X + 1$	15	0.9	2		X	
$X^4 + X + 1$	13	0.8	3	X		
$X^4 + X^2 + X + 1$	7	0.8	2	X	X	$X^3 + X + 1$
$X^4 + X^3 + X^2 + X + 1$	5	0.9	2,3,5,7	X		
$X^5 + X + 1$	21	0.8	2	X		
$X^5 + X + 1$	24	0.9	5		X	
$X^5 + X^4 + X^3 + X^2 + X + 1$	6	0.9	2,3,5,7	X		
$X^5 - X^3 - X^2 + X - 1$	8	0.9	3			$X^3 + X + 1$
$X^6 + X + 1$	31	0.8	5	X		
$X^6 + X^5 + X^4 + X^3 + X^2 + X + 1$	7	0.9	2,3,5,7	X		
$X^7 + X + 1$	48	0.9	7		X	
$X^7 + X^3 + X + 1$	15	0.9	2	X	X	
$X^7 + X^6 + \dots + X^2 + X + 1$	8	0.9	2,3,5,7	X		
$X^8 + X + 1$	63	0.9	2		X	
$X^8 + X + 1$	57	0.8	7	X		
$X^8 + X^4 + X^2 + X + 1$	15	0.9	2	X	X	$X^7 + X^3 + X + 1$
$X^8 + \dots + X^2 + X + 1$	9	0.9	2,3,5,7	X		
$X^9 + X + 1$	73	0.9	2	X		
$X^9 + X + 1$	80	0.9	3		X	
$X^9 + X^3 + X + 1$	26	0.9	3		X	
$X^9 - X^6 - X^5 + X^3 - X^2 + X - 1$	13	0.9	3			$X^4 + X + 1$
$X^9 + \dots + X^2 + X + 1$	10	0.9	2,3,5,7	X		
$X^{10} + X + 1$	91	0.9	3	X		
$X^{10} + \dots + X^2 + X + 1$	11	0.9	2,3,5,7	X		

$f$	$n$	$\beta$	$p$	T1	T2	T3
$X^{11} + X^8 + X^7 + X^5 + X^3 + X^2 + X + 1$	15	0.9	2	X		$X^4 + X + 1$
$X^{11} + \dots + X^2 + X + 1$	12	0.9	2,3,5,7	X		
$X^{12} + \dots + X^2 + X + 1$	13	0.9	2,3,5,7	X		
$X^{13} + X^4 + X + 1$	40	0.9	3	X		
$X^{13} + \dots + X^2 + X + 1$	14	0.9	2,3,5,7	X		
$X^{14} + \dots + X^2 + X + 1$	15	0.9	2,3,5,7	X		
$X^{15} + X^7 + X^3 + X + 1$	31	0.9	2	X	X	
$X^{15} + X^{14} + \dots + X^2 + X + 1$	16	0.9	2,3,5,7	X		
$nX^{16} + X + 1$	255	0.9	2		X	
$X^{16} + X^4 + X + 1$	63	0.9	2		X	
$X^{16} + X^8 + X^4 + X^2 + X + 1$	31	0.9	2	X	X	$X^{15} + X^7 + X^3 + X + 1$
$X^{16} + X^{12} + X^{11} + X^8 + X^6 + X^4 + X^3 + X^2 + X + 1$	21	0.9	2	X		$X^5 + X + 1$
$X^{16} + \dots + X^2 + X + 1$	17	0.9	2,3,5,7	X		
	$\vdots$					

Classification of the outputs of the algorithm

In order to have an efficient computation we used Sage to search linearized polynomial with the values of coefficients being included in  $\{0, 1, -1\}$ . Therefore the previous list is not exhaustive of all the equivalence classes for  $n' < 16$  when  $p$  is bigger than 3. For example, when we launch the algorithm for very small  $n$  with the coefficients allowed to be anything in  $\mathbb{F}_p$ , we get for  $p = 5$  and  $n = 4$ ,  $L_{(X^2+X+3)}$ . Indeed  $(X^2 + X + 3)(X^2 - X + 3) = (X^2 + 3)^2 - X^2 = X^4 + X^2 - 1 - X^2 = X^4 - 1$ , so  $X^2 + X + 3$  splits in  $\mathbb{F}_{5^4}$  and  $\beta = 4 * 1/4 = 1$ . Moreover, computing its equivalence class using Proposition 3.6, we observe that none element of its equivalence class has all its coefficients in  $\{0, 1, -1\}$ .

### 3.3 Linearized quasi-subfield polynomials with $n$ Mersenne

For  $p = 2$ ,  $p^{d+1} - 1$  is a Mersenne number . Therefore, type 2 gives examples of linearized quasi-subfield polynomials with  $n$  Mersenne. Therefore it may seem interesting to recall what was written in the appendix dedicated to linearized quasi-

subfield polynomials with  $n$  a Mersenne prime, present in the original article [11]. There, they were studied because of the fact that when  $n = 2^k - 1$  is a Mersenne prime,  $(X^n - 1)/(X - 1)$  has  $(n - 1)/k$  irreducible factors of degree  $k$  over  $\mathbb{F}_2$ , which gives a great number of potential candidates for linearized quasi-subfield polynomials in  $\mathbb{F}_{2^n}$ . However, because of a heuristic that we will recall here, [11] decides to exclude them from the promising leads to find linearized quasi-subfield polynomials.

**The reasoning of the original article** Let us consider  $k$  such that  $n = 2^k - 1$  is prime and denote  $N(k, n')$  the number of distinct polynomials of degree  $n'$  that divide  $X^n - 1$ .

Then [11] gives the following lemma:

**Lemma 3.8.** 
$$N(k, n') = \begin{cases} \binom{\lfloor n/k \rfloor}{\lfloor n'/k \rfloor} & \text{if } n' \bmod k = 0 \text{ or } 1 \\ 0 & \text{else} \end{cases}$$

Moreover,  $\log\left(\binom{\lfloor n/k \rfloor}{\lfloor n'/k \rfloor}\right) \simeq (n'/k) \log(n/n')$ .

Let us also introduce the heuristic used in [11]: For  $n$  a Mersenne prime, we may assume that the density of “sparse enough” polynomials (ie polynomials of the shape  $X^{p^{n'}} - \lambda(X)$  with  $\deg(\lambda) = \ell$ ) is identical for factors of  $X^n - 1$  and for random polynomials, for a value of  $\ell$  that we will precise later.

Since in  $\mathbb{F}_2[X]$ , there are  $2^{n'}$  monic polynomials of degree  $n'$  and  $2^\ell$  polynomials of degree  $\ell$ , this assumption allows us to approximate the number of polynomials of degree  $n'$  that divide  $X^n - 1$  and are sparse enough by  $N(k, n')2^{\ell - n'}$ . Therefore such polynomials a priori exist only if  $\ell > n' - (n'/k) \log(n/n')$ .

The case considered in the appendix of the article is when the quasi-subfield polynomials beat generic algorithms, which by Lemma 2.3 requires  $\alpha_\beta = \frac{1}{2c_{algo}\beta} \geq 1$ . Let us present it first even if the Type 2 does not fall in this category since its  $\alpha_\beta \simeq \frac{1}{2c_{algo}}$  is not bigger than 1. To improve on generic algorithms, we want  $\alpha_\beta = \frac{1}{2c_{algo}\ell n/n'^2} \geq 1$  hence

$\ell \leq \frac{n'^2}{2c_{algo}n}$ . With the previous constraint on  $\ell$ , it gives:  $\frac{n'^2}{2c_{algo}n} > n' - (n'/k) \log(n/n')$ . Thus, since  $k \simeq \log(n)$ , we get  $\frac{n'}{2c_{algo}n} > 1 - \log(n/n')/\log(n) = \log(n')/\log(n)$ . Therefore,  $\frac{\log(n)}{2c_{algo}n} < \frac{\log(n')}{n'}$ .

Since  $2c_{algo} \simeq 10$ , and  $n' < n$ , this inequality fails and with the heuristic, there should not be any linearized quasi-subfield polynomials with  $n$  Mersenne beating generic algorithms.

**Our adaptation of this reasoning to the case of Type 2** Type 2 polynomials are quasi-subfield polynomials, so they respect  $\beta = \ell.n/n'^2 \leq 1$ . Therefore, we have  $\ell \leq n'^2/n$ . This constraint added to the same heuristic as before gives:  $\frac{n'^2}{n} > n' - (n'/k) \log(n/n')$  which similarly as in the previous paragraph gives  $\frac{\log(n)}{n} < \frac{\log(n')}{n'}$ . Since  $x \mapsto \log(x)/x$  is decreasing for  $x > e$ , this is also not possible for  $n > n' \geq 3$ .

Therefore, the Type 2 polynomials provide a counter-example to the heuristic. Hence, we may deduce that when  $n$  is a Mersenne prime, there exists an  $\ell$  such that the density of “sparse enough” polynomials (ie polynomials of the shape  $X^{p^{n'}} - \lambda(X)$  with  $\deg(\lambda) = \ell$ ) is bigger for factors of  $X^n - 1$  than for random polynomials.

## 4 Lower bounds on $\beta$ for linearized quasi-subfield polynomials

We will now introduce one of our main results: a lower bound on  $\beta$  for linearized quasi-subfield polynomials. We looked for a result of this type after observing that for all the quasi-subfield polynomial returned by our systematic search using Sage (cf Appendix B),  $\beta$  was always equal or bigger than  $3/4$ . The equality is obtained for  $X^{p^2} + X^p + X$  in  $\mathbb{F}_{p^3}[X]$  (Type 1bis) and all its equivalence class.

### 4.1 The result

**Theorem 4.1** ( $\beta \geq 3/4$ ). Let  $L = X^{p^{n'}} - (a_\ell X^{p^\ell} + a_{\ell-1} X^{p^{\ell-1}} + \dots + a_0 X)$  a linearized polynomial with  $\ell \geq 1$  and  $\forall i, a_i \in \mathbb{F}_{p^n}$ . If  $L$  completely splits over  $\mathbb{F}_{p^n}$  then  $\beta = \ell.n/n'^2 \geq 3/4$ .

This result is in fact a consequence of the following lemma which highlights that the field has to be big enough to have completely splitting linearized polynomials in it.

**Lemma 4.2** (Lower bound on  $n$ ). Let  $L = X^{p^{n'}} - (a_\ell X^{p^\ell} + a_{\ell-1} X^{p^{\ell-1}} + \dots + a_0 X)$  with  $\ell \geq 1$  and  $\forall i, a_i \in \mathbb{F}_{p^n}$ . If  $L$  completely splits over  $\mathbb{F}_{p^n}$  then

$$n \geq n' + (n' - \ell) \left\lfloor \frac{n' - 1}{\ell} \right\rfloor$$

Before proving this lemma, we will begin by proving that this result is enough to prove the theorem.

*Proof.* One can first observe that since  $n'$  and  $\ell$  are integers:  $\left\lfloor \frac{n' - 1}{\ell} \right\rfloor \geq \frac{n'}{\ell} - 1$ .

Therefore, by the lemma 4.2,  $n \geq n' + (n' - \ell) \frac{n'}{\ell} - (n' - \ell) \geq \frac{n'^2}{\ell} - n' + \ell$

Thus  $\beta = \frac{\ell n}{n'^2} \geq 1 - \frac{\ell}{n'} + \frac{\ell^2}{n'^2} = 1 - \frac{\ell}{n'} \left(1 - \frac{\ell}{n'}\right)$ .

Since the maximum of  $x \in [0, 1] \mapsto x(1 - x)$  is  $1/4$ , we have  $\beta \geq 3/4$ .  $\square$

Let us now reduce the proof of the lemma to the proof of a result about the power of matrix. Property 3.3 gives that  $L$  completely splits over  $\mathbb{F}_{p^n}$  if and only if  $A_L = C_L \cdot C_L^\sigma \dots C_L^{\sigma^{n-1}} = I$

$$\text{with } C_L = \begin{bmatrix} 0 & 0 & \dots & 0 & \dots & \dots & -a_0 \\ 1 & 0 & \dots & 0 & \dots & \dots & -a_1 \\ 0 & 1 & \dots & 0 & \dots & \dots & -a_2 \\ \vdots & \vdots & \ddots & & \ddots & & \vdots \\ 0 & 0 & \dots & 1 & \dots & \dots & -a_\ell \\ \vdots & \vdots & & \vdots & \ddots & & 0 \\ \vdots & \vdots & & \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & & \dots & 1 & 0 \end{bmatrix} \text{ a matrix of size } n' \times n'.$$

Therefore we have to prove that

$$\forall r < n' + (n' - \ell) \left\lfloor \frac{n' - 1}{\ell} \right\rfloor, C_{L,r} := C_L \cdot C_L^\sigma \dots C_L^{\sigma^{r-1}} \neq I$$

Our result depends only of the value of  $n'$  and  $\ell$ , therefore we will abstract a little more the values of  $C_L$ : to do this, we attribute to each coefficient of  $C_L$  a symbol: 0 if for any  $p$  and any values of the  $(a_i)_{i \leq \ell}$  this coefficient is zero, 1 for 1,  $pow(x)$  for the powers of  $x = (-a_\ell) \neq 0$ , and  $\blacksquare$  for anything else. Knowing that  $x \neq 0$ , all the coefficients marked  $pow(x)$  are non null.

The operations on these categories are as follow:

+	0	1	$pow(x)$	$\blacksquare$
0	0	1	$pow(x)$	$\blacksquare$
1	1	$\blacksquare$	$\blacksquare$	$\blacksquare$
$pow(x)$	$pow(x)$	$\blacksquare$	$\blacksquare$	$\blacksquare$
$\blacksquare$	$\blacksquare$	$\blacksquare$	$\blacksquare$	$\blacksquare$

$\times$	0	1	$pow(x)$	$\blacksquare$
0	0	0	0	0
1	0	1	$pow(x)$	$\blacksquare$
$pow(x)$	0	$pow(x)$	$pow(x)$	$\blacksquare$
$\blacksquare$	0	$\blacksquare$	$\blacksquare$	$\blacksquare$

Moreover, recalling that  $\sigma$  acts on matrices coefficient-wise, one can observe that



all the  $C_L^{\sigma^k}$  can be written as

$$M := \begin{bmatrix} 0 & 0 & \dots & 0 & \dots & \dots & \blacksquare \\ 1 & 0 & \dots & 0 & \dots & \dots & \blacksquare \\ 0 & 1 & \dots & 0 & \dots & \dots & \blacksquare \\ \vdots & \vdots & \ddots & & \ddots & & \vdots \\ 0 & 0 & \dots & 1 & \dots & \dots & \text{pow}(x) \\ \vdots & \vdots & & \vdots & \ddots & & 0 \\ \vdots & \vdots & & \vdots & \ddots & & \vdots \\ 0 & 0 & \dots & \dots & & 1 & 0 \end{bmatrix}$$

Therefore with this notation,  $C_{L,r}$  is merely  $M^r$ . Our goal is then to study the shape of the powers of  $M$ , which is a companion matrix defined on  $\{0, 1, \text{pow}(x), \blacksquare\}$ .

**Lemma 4.3** (The powers are not the identity).

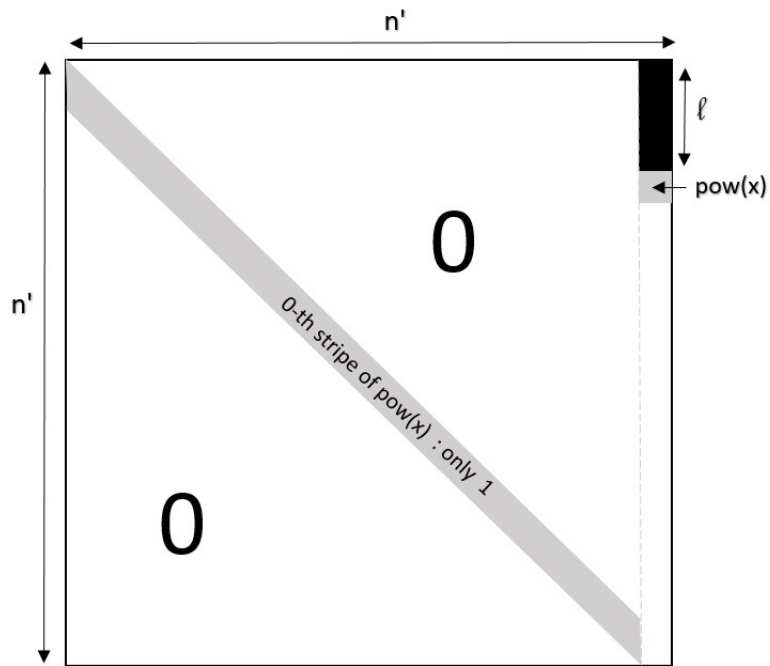
$$\forall r < n' + (n' - \ell) \left\lfloor \frac{n' - 1}{\ell} \right\rfloor, M^r \neq I$$

The two next subsections will be dedicated to proofs of this lemma. The first one will give an informal proof based on visual intuition whereas the second one will be a more formal proof.

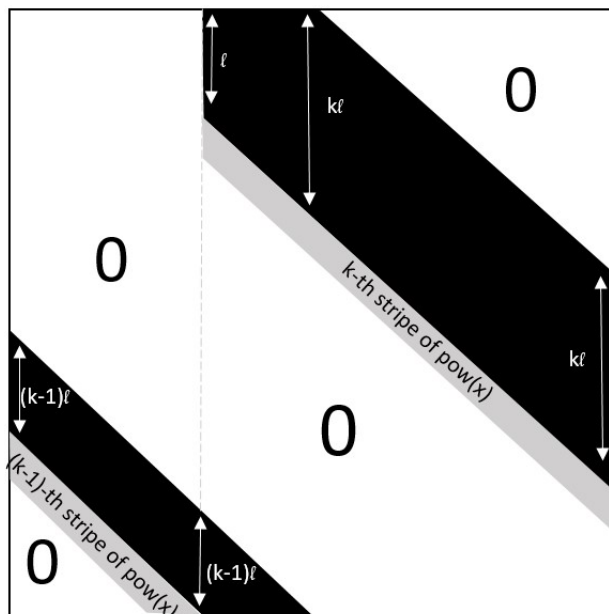
## 4.2 The informal proof

Let us first give an informal proof of this claim: We will use a visual representation of the matrices. We will represent in white the zero coefficients in the matrices, grey the powers of  $x$  (including  $x^0 = 1$ ) and black all the coefficients for which we cannot track the value.

For example,  $M$  is represented by:



Moreover, one can convince itself that the shape of a  $M^r$  is (for some values of  $l, n'$  and  $r$ , three grey stripes may appear):



Then multiplying by  $M$  implies a shift of the stripes toward the bottom-left corner. Therefore stripes appear at the right side and disappear at the left side when computing the different  $M^r$ . Every time a new stripe appears, it is larger and thus progressively the whole matrix is covered by the black - unknown coefficients.

When this happens, we have completely lost track of the value of the coefficients of the matrix. From this time, the powers of  $M$  may happen to be the identity but we are unable to say when. However, as long as the grey part of the stripes is present, we are able to prove that the matrix is not the identity by showing that a coefficient of this grey stripe is not on the main diagonal of the matrix.

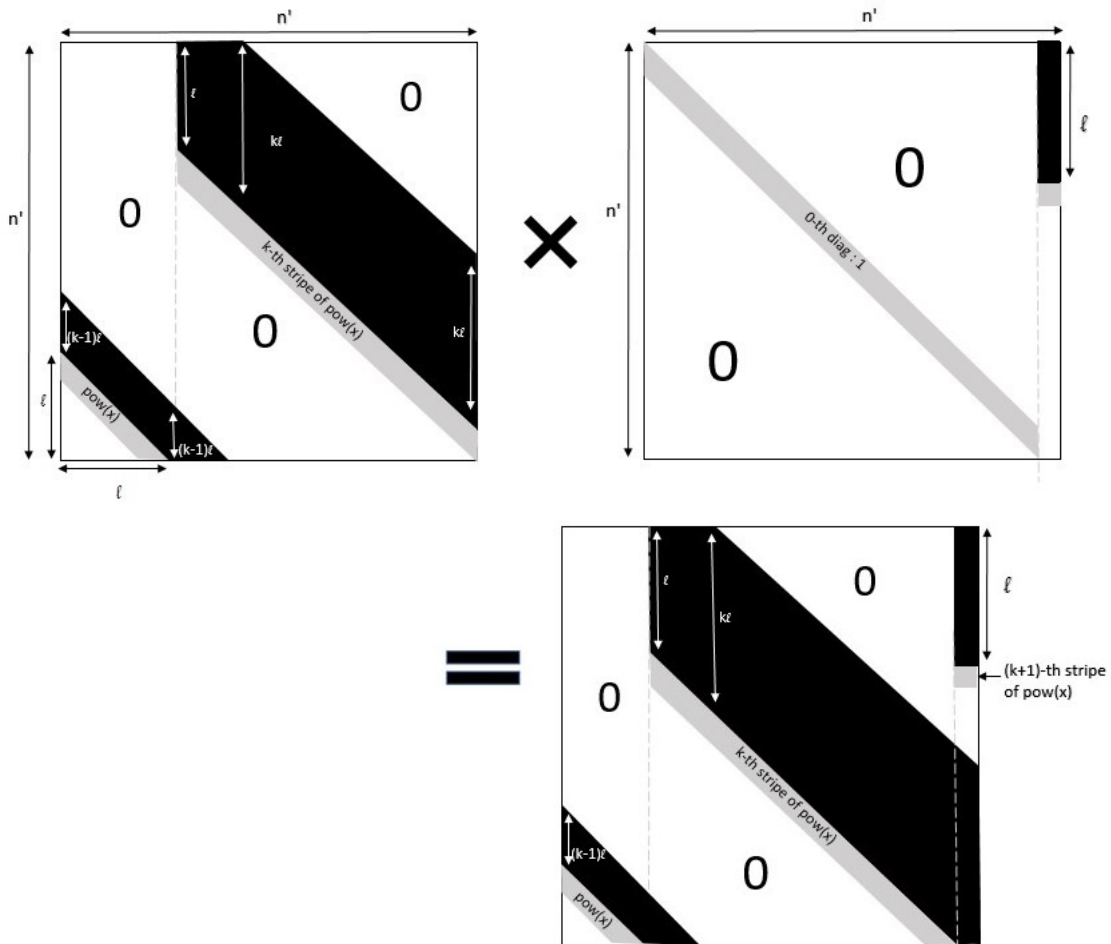
Therefore we need to count the stripes of  $pow(x)$  appearing in the successive  $M^r$  (since the width of the stripe depend of its ranking in the order of apparition) and know until which  $r$ , the  $k^{\text{th}}$  stripe of  $pow(x)$  is present in the matrix  $M^r$ .

Therefore we will search the answer to the following questions:

1. At what condition the  $(k + 1)^{\text{th}}$  stripe of  $pow(x)$  appear?
2. When does the  $k^{\text{th}}$  stripe of  $pow(x)$  disappear (call this time  $D_k$ )?
3. What is the largest  $R$  where we are sure that for all  $1 \leq r \leq R$ ,  $M^r \neq I$ ?

1. **At what condition does the  $(k + 1)^{\text{th}}$  stripe of  $pow(x)$  appear?**

One can observe that the following operation leads to the apparition of the first cell of the  $(k + 1)^{\text{th}}$  stripe of  $pow(x)$ :



However, this operation is possible if and only if there are only 0 at the left of the first coefficient of the  $k^{\text{th}}$  stripe of  $pow(x)$ . We do not know what is in the  $(k-1)$ -th black stripe, so we will just give a sufficient condition for this to happen: the  $(k-1)$ -th black stripe begins after the  $(\ell + 1)$ -th row of the matrix. Looking at the composition of the first column we see that this is true when  $\ell + (k - 1)\ell \leq n' - (\ell + 1)$  ie  $k + 1 \leq \frac{n' - 1}{\ell}$ .

**2. When does the  $k^{\text{th}}$  stripe of  $pow(x)$  disappear?**

Let us consider  $k \leq \frac{n' - 1}{\ell}$  so that the  $k^{\text{th}}$  stripe of  $pow(x)$  appears in some  $M^r$ . We will denote  $D_k$  the time where this stripe disappears.

At time  $D_k$ , the  $(k + 1)^{\text{th}}$  stripe of  $pow(x)$  begins at the  $(\ell + 1)$  row of the first column, and each time we multiply by  $M$  all the stripes move to one column to the left, so one can observe that  $D_{k+1} = D_k + (n' - \ell)$  and thus  $D_k = n' + k(n' - \ell)$ .

**3. What is the largest  $R$  where we are sure that for all  $1 \leq r \leq R$ ,  $M^r \neq I$ ?**

The  $(k + 1)^{\text{th}}$  stripe of  $pow(x)$  appears when  $k + 1 \leq \frac{n' - 1}{\ell}$ . Therefore, the last stripe of  $pow(x)$  that we are sure will be present is the  $k_{\text{max}}^{\text{th}}$  one with  $k_{\text{max}} = \left\lfloor \frac{n' - 1}{\ell} \right\rfloor$ .

It disappears at time  $D_{k_{\text{max}}} = n' + (n' - \ell)k_{\text{max}}$ , therefore for all  $r < D_{k_{\text{max}}}$ ,  $M^r$  is of the previously shown shape.

Let us now show that for  $r < D_{k_{\text{max}}}$ ,  $M_{i_r, 1}^r = pow(x) \neq 0$  with

$$i_r = r - (n' - \ell) \left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor + 1$$

Indeed, one can notice that at time  $D_k$ , with  $k < k_{\text{max}}$ , the  $(k + 1)^{\text{th}}$  stripe of  $pow(x)$  begins at the  $(\ell + 1)$  row of the first column, ie  $M_{\ell+1, 1}^{(D_k)} = pow(x) \neq 0$ . Each time we multiply by  $M$  all the stripes move to one column to the left, so for any  $r$  such that  $D_k \leq r < D_{k+1}$  we write  $r = D_k + a$  and we have  $M_{\ell+1+a, 1}^r = pow(x) \neq 0$ .

Let us now consider a generic integer  $r < D_{k_{\text{max}}}$ . Then, noting  $k = \left\lfloor \frac{r - n'}{n' - \ell} \right\rfloor$ , we have  $D_k = n' + (n' - \ell) \left\lfloor \frac{r - n'}{n' - \ell} \right\rfloor \leq r < D_{k+1}$ . Thus  $M_{\ell+1+a, 1}^r = pow(x) \neq 0$  with  $a = r - D_k = r - (n' - \ell) \left\lfloor \frac{r - n'}{n' - \ell} \right\rfloor - n'$ .

$$\begin{aligned}
\text{Moreover, } \ell + a + 1 &= r - (n' - \ell) \cdot \left\lfloor \frac{r - n'}{n' - \ell} \right\rfloor - n' + \ell + 1 \\
&= r - (n' - \ell) \cdot \left( \left\lfloor \frac{r - n'}{n' - \ell} \right\rfloor + 1 \right) + 1 \\
&= r - (n' - \ell) \left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor + 1
\end{aligned}$$

Hence,  $M_{i_r,1}^r = \text{pow}(x) \neq 0$  with  $i_r = r - (n' - \ell) \left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor + 1$ . Moreover,  $\ell \geq 1$ , we have that  $i_r = \ell + a + 1 > 1$ . This means that there is a non zero coefficient on the first column which is not on the first row in  $M^r$ , hence  $M^r \neq I$ .

Therefore, the largest  $R$  where we are sure that for all  $1 \leq r \leq R$ ,  $M^r \neq I$  is  $R = D_{k_{\max}} = n' + (n' - \ell) \left\lfloor \frac{n' - 1}{\ell} \right\rfloor$  which is exactly the value we wanted to find to prove the lemma 4.3.

### 4.3 The formal proof

Let us now give a more formal proof of lemma 4.3. For this, we will use the result of *The combinatorial power of the companion matrix* [4], by Chen and Louck, about the powers of companion matrices. After a few modifications to make their results match our definition of companion matrices (see Appendix A.1.2), we have:

$$M_{i,j}^r = \begin{cases} 1 & \text{if } r = i - j \\ \sum_{k_1, \dots, k_{n'}} w_{\mathbf{k}} 0^{k_1 + \dots + k_{n'-\ell-1}} \text{pow}(x)^{k_{n'-\ell} \blacksquare^{k_{n'-\ell} + \dots + k_{n'}}} & \text{otherwise} \end{cases}$$

where  $\mathbf{k} = (k_i)_{1 \leq i \leq n'}$  are non-negative integers such that  $k_1 + 2k_2 + \dots + n'k_{n'} = r - i + j$  and  $w_{\mathbf{k}} = \frac{k_j + \dots + k_{n'}}{k_1 + \dots + k_{n'}} \binom{k_1 + \dots + k_{n'}}{k_1, \dots, k_{n'}}$

If  $k_1 + \dots + k_{n'-\ell-1} > 0$  then  $0^{k_1 + \dots + k_{n'-\ell-1}} \text{pow}(x)^{k_{n'-\ell} \blacksquare^{k_{n'-\ell} + \dots + k_{n'}} = 0$ , so we can remove all terms involving a positive  $k_i$  with  $i < n' - \ell$ . Therefore we get

$$M_{i,j}^r = \begin{cases} 1 & \text{if } r = i - j \\ \sum_{k_{n'-\ell}, \dots, k_{n'}} w_{\mathbf{k}} \text{pow}(x)^{k_{n'-\ell} \blacksquare^{k_{n'-\ell} + \dots + k_{n'}}} & \text{otherwise} \end{cases}$$

where  $(n' - \ell)k_{n'-\ell} + \dots + n'k_{n'} = r - i + j$ . Let us observe that  $w_{\mathbf{k}}$  is now

$$w_{\mathbf{k}} = \frac{k_{\max(n'-i+1, n'-\ell)} + \dots + k_{n'}}{k_{n'-\ell} + \dots + k_{n'}} \binom{k_{n'-\ell} + \dots + k_{n'}}{k_{n'-\ell}, \dots, k_{n'}}$$

We do not want to remove the exponents over the black square. Indeed when  $k_{n'-\ell} + \dots + k_{n'} = 0$ , we have  $\blacksquare^{k_{n'-\ell} + \dots + k_{n'}} = 1$ ; so we get a term  $w_{\mathbf{k}} \text{pow}(x)^{k_{n'-\ell}}$  which in the case  $w_{\mathbf{k}} \neq 0$  is exactly what we want to prove the lemma 4.3.

The first line of this expression of the coefficients of  $M^r$  can only be used when  $r < n'$ . Therefore, let us split the proof of the lemma 4.3 in two claims according to whether  $r < n'$ .

Claim 1:

$$\forall r < n', M_{r+1,1}^r = 1.$$

This is obvious thanks to the previous expression of the powers of  $M$ . Indeed,  $(r+1) - 1 = r$ .

Claim 2:

$$\begin{aligned} \forall n' \leq r < n' + (n' - \ell) \left\lfloor \frac{n' - 1}{\ell} \right\rfloor, M_{i_r,1}^r = \text{pow}(x) \\ \text{with } i_r = r - (n' - \ell) \left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor + 1 \in [2, n'] \end{aligned}$$

One can notice that  $i_r$  is inspired by the value found in the informal proof.

Let  $n' \leq r \leq n' + (n' - \ell) \left\lfloor \frac{n' - 1}{\ell} \right\rfloor - 1$ . Let us first observe that  $M_{i_r,1}^r$  is well defined and is not in the top left corner since:

$$\begin{aligned} i_r = r - (n' - \ell) \left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor + 1 &\leq r - (n' - \ell) \left( \frac{r - \ell}{n' - \ell} - \frac{n' - \ell - 1}{n' - \ell} \right) + 1 \\ &\leq r - (r - n' + 1) + 1 \\ &\leq n' \end{aligned}$$

and

$$\begin{aligned} i_r = r - (n' - \ell) \left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor + 1 &\geq r - (n' - \ell) \frac{r - \ell}{n' - \ell} + 1 \\ &\geq \ell + 1 \\ &\geq 2 \end{aligned}$$

$$\text{Moreover, } \frac{r - \ell}{n' - \ell} \leq \frac{n' + (n' - \ell) \left\lfloor \frac{n' - 1}{\ell} \right\rfloor - 1 - \ell}{n' - \ell} \leq \left\lfloor \frac{n' - 1}{\ell} \right\rfloor + \left( 1 - \frac{1}{n' - \ell} \right).$$

Then  $1 \leq \left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor \leq \left\lfloor \frac{n' - 1}{\ell} \right\rfloor$ . Furthermore,  $n' - i_r + 1 \leq n' - (\ell + 1) + 1 = n' - \ell$  so  $\max(n' - i_r + 1, n' - \ell) = n' - \ell$ .

$$\begin{aligned} \text{So } M_{i_r,1}^r &= \sum_{k_{n'-\ell}, \dots, k_{n'}} \frac{k_{n'-\ell} + \dots + k_{n'}}{k_{n'-\ell} + \dots + k_{n'}} \binom{k_{n'-\ell} + \dots + k_{n'}}{k_{n'-\ell}, \dots, k_{n'}} \text{pow}(x)^{k_{n'-\ell} \blacksquare k_{n'-\ell} + \dots + k_{n'}} \\ &= \sum_{k_{n'-\ell}, \dots, k_{n'}} \binom{k_{n'-\ell} + \dots + k_{n'}}{k_{n'-\ell}, \dots, k_{n'}} \text{pow}(x)^{k_{n'-\ell} \blacksquare k_{n'-\ell} + \dots + k_{n'}} \end{aligned}$$

where  $(n' - \ell)k_{n'-\ell} \dots + n'k_{n'} = r - i_r + 1 = (n' - \ell) \left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor$  (1)

$$k_{n'-\ell+1} = \dots = k_{n'} = 0 \text{ and } k_{n'-\ell} = \left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor \text{ verifies (1).}$$

Let us show that there are no other solutions. Let us assume that there exist  $k_{n'-\ell}, \dots, k_{n'}$  such that  $\sum_{i=n'-\ell}^{n'} i \cdot k_i = (n' - \ell) \left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor$  and  $\sum_{i=n'-\ell+1}^{n'} k_i > 0$ .

$$\begin{aligned} \text{Then, } (n' - \ell) \left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor &= \sum_{i=n'-\ell}^{n'} i \cdot k_i \\ &= (n' - \ell) \sum_{i=n'-\ell}^{n'} k_i + \sum_{i=n'-\ell+1}^{n'} (i - (n' - \ell)) k_i \\ &= (n' - \ell) \sum_{i=n'-\ell}^{n'} k_i + \sum_{i=1}^{\ell} i k_{n'-\ell+i} \end{aligned}$$

Since  $\sum_{i=1}^{\ell} k_{n'-\ell+i} > 0$ , we have  $\sum_{i=1}^{\ell} i k_{n'-\ell+i} > 0$  and  $(n' - \ell) \mid \sum_{i=1}^{\ell} i k_{n'-\ell+i}$ . Therefore  $\sum_{i=1}^{\ell} i k_{n'-\ell+i} \geq n' - \ell$ .

$$\begin{aligned} \text{Thus, } (n' - \ell) \left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor &\geq (n' - \ell) (1 + \sum_{i=n'-\ell}^{n'} k_i) \\ &\geq (n' - \ell) \left( 1 + \frac{\sum_{i=1}^{\ell} i k_{n'-\ell+i}}{\ell} \right) \\ &\geq (n' - \ell) \left( 1 + \frac{n' - \ell}{\ell} \right) \\ &\geq (n' - \ell) \cdot \frac{n'}{\ell} \end{aligned}$$

Since  $\left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor \leq \left\lfloor \frac{n' - 1}{\ell} \right\rfloor < \frac{n'}{\ell}$ , this is absurd.

$$\text{Therefore } M_{i_r, 1}^r = \binom{k_{n'-\ell}}{k_{n'-\ell}} \text{pow}(x) \left\lfloor \frac{r - \ell}{n' - \ell} \right\rfloor = \text{pow}(x)$$

Hence,  $M^r \neq I_{n'}$

#### 4.4 Comparison with other lower bounds

In the original paper, a lower bound was given in the lemma 4.1: for all  $L$  quasi-subfield polynomial,

$$\left\lfloor \frac{n}{n'} \right\rfloor \ell + (n \bmod n') \geq n'$$

Therefore  $\beta \simeq \left\lfloor \frac{n}{n'} \right\rfloor \frac{\ell}{n'} \geq 1 - \frac{(n \bmod n')}{n'}$ . Depending on the value of  $\frac{(n \bmod n')}{n'}$ , the bound given here is sharper or not. Heuristically, if  $(n \bmod n')$  is considered as uniform at random between 0 and  $n' - 1$ , then there is only a probability of approximately 1/4 to have the lower bound  $\beta \geq 3/4$ . This bound is given by Theorem 1.2 without any heuristic involved. However it is only valid when  $L$  is a linearized polynomial.

Moreover, this bound is similar to the one given by Daniela Mueller and Gary McGuire in [12] which was established during the completion of this dissertation. Indeed, using also [13] they have shown (in Theorem 1.1) that for any linearized trinomial  $L = X^{q^d} - bX^q - aX \in \mathbb{F}_{p^n} = \mathbb{F}_{q^{\tilde{n}}}$  with  $b \neq 0$ , the fact that it completely

splits implies that  $\tilde{n} \geq (d-1)d + 1 = d^2 - d + 1$ . Let us compare it with the bound given by our lemma 4.2. Let us write  $n = k\tilde{n}$  so that  $q = p^k$ . Thus  $L = X^{p^{kd}} - bX^{p^k} - aX$ . The lemma gives:  $n \geq kd + (kd - k) \left\lfloor \frac{kd - 1}{k} \right\rfloor$  so  $k\tilde{n} \geq k(d + (d-1)(d-1))$ . Thus  $\tilde{n} \geq d + (d-1)^2 = d^2 - d + 1$ . Therefore, we obtain exactly the same bound.

Even if their bound is less general than ours by only considering trinomials, they gave a more complete description of completely splitting linearized trinomials. Indeed, this description takes into account the case  $\ell = 0$  which we did not consider in this dissertation, and describes exhaustively the different possibilities:

- either  $\tilde{n} = id$ ,  $b = 0$  and  $a^{1+q^d+\dots+q^{(i-1)d}} = 1$
- either  $\tilde{n} = (d-1)d + 1$ ,  $a^{1+q+\dots+q^{(d-1)d}} = (-1)^{d-1}$ ,  $b = -a^{qe_1}$  where  $e_1 = \sum_{i=0}^{d-1} q^{id}$  and  $d-1$  is a power of  $p$
- either  $\tilde{n} > (d-1)d + 1$

## 4.5 Consequences of this theorem

Let us now study the consequences of the Theorem 1.2. Let  $L$  be a linearized quasi-subfield polynomial. Then  $\beta(L) \geq 3/4$  and  $\alpha_\beta = \frac{1}{2 \cdot c_{algo} \beta(L)} \leq \frac{2}{3 \cdot c_{algo}} < 1/7$ . Hence  $a_\beta \geq 1$  is false and then it is not possible to beat generic algorithms with  $L$ . The best complexity we can hope is indeed  $\tilde{O}(p^{(1-\alpha_\beta(1/2-1/m))n})$  which is bigger than  $\tilde{O}(p^{(1-1/14)n})$ .

To be more precise, the previous estimation used the approximation of  $c_{algo} \simeq 4.876$ . If we succeeded to have  $c_{algo} < 1.5$  then, when  $\beta(L) = 3/4$ , we would have  $\alpha_\beta > 1$ , so such a polynomial  $L$  could allow us to have an algorithm running faster than generic algorithms. However, this drastic reduction of  $c_{algo}$  does not seem plausible. Therefore, we decided not to consider the track of establishing a finer estimate of the complexity of the algorithm introduced in 2.

This questions the strategy used so far to find efficient quasi-subfield polynomials because the path of using the linearized polynomial of a polynomial dividing  $X^n - 1$  is now conditional to a major improvement in the estimation of  $c_{algo}$ . However, it does not mean that there are no quasi-subfield polynomial allowing to beat generic algorithms with the current estimation of  $c_{algo}$ , but if there exists one, then it is not a linearized polynomial.



As the roots of any linearized polynomial form an additive group, we can be interested in what happen with other groups. Therefore, we will now study what happen when we consider a multiplicative group.

## 5 Use of multiplicative groups

Let us now follow an other track suggested by [11] to find quasi-subfield polynomials whose roots form a multiplicative group. To do this, we will first recall the conditions that it implies on the coefficients, then provide three new families of quasi-subfield polynomials which verifies these conditions. Finally, we will discuss the fact that these polynomials do not split completely but still can be used in the algorithm to solve the ECDLP.

### 5.1 Polynomials based on multiplicative groups

In the original article, the use of quasi-subfield polynomials of the type  $L = X^{p^{n'}} - X^a$  with  $a = p^{n'} \bmod r$ ,  $r|p^n - 1$  and  $n' > \log_p(r)$  is suggested. They are obtained from the multiplicative group formed by the solutions of  $X^r - 1$ . We can factor  $L$  as  $L = X^a(X^{p^{n'}-a} - 1)$  so the maximum number of roots of  $L$  in  $\mathbb{F}_{p^n}$  is  $1 + p^{n'} - a$ . But one knows that there are  $\gcd(k, p^n - 1)$  roots of  $X^k - 1$  in  $\mathbb{F}_{p^n}$ . Hence we are looking for tuples  $(p, n, n', r)$  such that for  $a := p^{n'} \bmod r$ , and  $\gcd(p^{n'} - a, p^n - 1) = p^{n'} - a$  ie  $p^{n'} - a|p^n - 1$ .

This matches a looser definition of a quasi-subfield polynomial: instead of requiring that the polynomial completely splits, we will require that it splits and has all its roots simple except 0. More information about the impact of this slight modification will be given in paragraph 5.3.

### 5.2 New families of quasi-subfield polynomials

Similarly as for the search of linearized quasi-subfield polynomials, we used SageMath to provide us a list of values of  $n'$  and  $r$  which fulfil the previously mentioned requirements and also verify  $n \cdot \log_p(a)/n'^2 \leq 1$  (the condition on  $\beta$  where  $a := p^{n'} \bmod r$ ). To do this it considers all the possible value for  $r$  among the divisors of  $p^n - 1$  and outputs only the one meeting the requirements. This outputs quite a messy set of polynomials (see A.2.2). However, using *he On-Line Encyclopedia of Integer Sequences*<sup>1</sup> [1], we identified the frequent presence of  $p$  of the shape  $k^n + k - 1$

<sup>1</sup>see <https://oeis.org/A002327> and <https://oeis.org/A100698> for examples

associated with  $n' = 1$  and of  $p$  of the shape  $k^n - k - (-1)^n$  with  $n' = n - 1$ . After some more work to conjecture the values of  $r$  associated in each case, we were able to establish three families of multiplicative quasi-subfield polynomials:

**Proposition 5.1** (Multiplicative quasi-subfield polynomials). Let us consider three possible sets of parameters

1. Let  $p$  prime and  $k \geq 2$  and  $i \geq 1$  integers. Let  $n = 2ik$ ,  $n' = i(2k - 1) = n - i$  and  $r = \frac{p^n - 1}{p^{2i} - 1}$ .
2. Let  $p = k^n + k - 1$  prime and  $k \geq 2$  an integer. Let  $n' = 1$  and  $r = (p - k)/(k - 1)$ .
3. Let  $p = k^n - k - (-1)^n$  be prime,  $n > 2$  and  $k > 1$  integers such that  $k^n \gg 1$ . Let  $n' = n - 1$  and  $r = \frac{(p^n - 1)(k - (-1)^n)}{(k^n - k)(k^n - (-1)^n)}$

Then for each set,  $a = p^{n'} \pmod r$  is such that  $L = X^{p^{n'}} - X^a$  is a quasi-subfield polynomial in  $\mathbb{F}_{p^n}$ .

Before proving this proposition, let us give a few remarks about these families.

- First, we can observe that contrary to the first family, the two other may be used with  $n$  prime. Therefore, they correspond to the most frequent case where we need to use the ECDLP in cryptography: a field  $\mathbb{F}_{p^n}$  with  $p$  and  $n$  prime.
- Secondly, one can notice that for the first family, with  $p = 2$ ,  $i = 1$  and  $k = 2$ , we get  $r = (2^4 - 1)/(2^2 - 1) = 5$  and  $a = 3$ , thus  $L = X^8 - X^3$  is a quasi-subfield polynomial. However, for this polynomial  $\beta = \log_2(3) * 4/3^2 \simeq 0.70 < 0.75$ . Hence, the previously proved lower bound on  $\beta$  is not valid for multiplicative quasi-subfield polynomials. It leads to an algorithm with a complexity less than  $O(p^{0.93n})$
- Thirdly, we did not prove the existence of a prime  $p$  of the shape  $k^n + k - 1$  or  $k^n - k - (-1)^n$  for any  $n$  prime. For some  $n$ , they indeed do not exist: for exemple with  $n = 5$ ,  $k^5 + k - 1 = (k^3 + k^2 - 1)(k^2 - k + 1)$  so  $k^5 + k - 1$  is not prime as soon as  $k > 1$ . Therefore, we highlight that the proposition only says that if such couple  $(p, n)$  exists then we can build a quasi-subfield polynomial in  $\mathbb{F}_{p^n}$ .

- Furthermore,  $(k-1)^2 - (k-1) - 1 = k^2 - k - 1$  so the last two families overlap when  $n = 2$ . We excluded the case  $n = 2$  in the last family, because such a choice of  $n'$  and  $r$  would lead to  $\beta = 0$  which is not allowed in our definition of quasi-subfield polynomials. However, thanks to the two last families, we have a multiplicative quasi-subfield polynomial for any  $n$  and  $p = k^n - k - (-1)^n$  prime.
- Last but not least, it is worth noticing that the case  $p = k^n - k - (-1)^n$  is the most promising among the families introduced. Indeed, primes of the form  $f(2^m)$ , where  $f(x)$  is a low-degree polynomial with small integer coefficients, are often used in cryptography since they were introduced in [17]. Indeed as well as for Mersenne primes, they allow fast modular reduction. They are called Solinas primes, or generalized Mersenne primes. Coming back to our exemple,  $f(x) = x^n - x - (-1)^n$  verifies the constraint required about the weights of the coefficients, so the last family when applied with  $k$  a power of 2 corresponds to Solinas primes. It is then important to notice that Curve448, which is part of the approved elliptic curves for use by the US Federal Government, uses a prime exactly of this shape:  $p = 2^{448} - 2^{224} - 1$ . [10][5]. Moreover, four others curves recommended by NIST in 1999 [14] also uses Solinas primes : p-192 ( $p = 2^{192} - 2^{64} - 1$ ), p-224 ( $p = 2^{224} - 2^{96} + 1$ ) and p-256 ( $p = 2^{256} - p^{224} + 2^{192} + 2^{96} - 1$ ) and p-384 ( $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$ ). Therefore, it may seem interesting to study more deeply multiplicative quasi-subfield polynomials when  $p$  is a Solinas prime. For a list of Solinas primes of the shape  $2^n - 2^m \pm 1$ , one can consult [17]. Of course, this approach is still far from threatening the security of these curves : they are defined on a prime field  $F_p$  (so the objective would be to have a complexity better than  $0(\sqrt{p})$ ) while we are considering an extension field  $F_{p^n}$  with  $n \geq 2$  and have  $\beta \simeq 1$  so we obtain a complexity of  $0(p^{0.95n})$ .

*Proof.* Let us now prove that these sets of parameters lead to quasi-subfield polynomials.

1. Since  $2i|n$ , it is obvious that  $r = \frac{p^n-1}{p^{2i}-1}$  is an integer an  $r|p^n - 1$ .

Let us first explicit the calculus of  $a$  by observing that

$$\begin{aligned} r.(p^i - 1) &= \frac{p^n-1}{p^{2i}-1} \cdot (p^i - 1) = \frac{p^{2ik}-1}{p^i+1} = p^{2ik-i} - \frac{p^{2ik-i}+1}{p^i+1} \\ &= p^{n'} - c \text{ with } c = \frac{p^{i(2k-1)}+1}{p^i+1} > 0 \end{aligned}$$

But,

$\frac{c}{r} = \frac{(p^{i(2k-1)+1})(p^{2i-1})}{(p^{i+1})(p^{2ik-1})} = \frac{(p^{i(2k-1)+1})(p^i-1)}{p^{2ik-1}} = 1 - \frac{p^{i(2k-1)}-p^i}{p^{2ik-1}} < 1$  since  $2k-1 > 1$ . Thus  $0 < c < r$ , and then  $a = (p^{n'} \bmod r) = c$ .

Therefore  $p^{n'} - a = r \cdot (p^i - 1) = \frac{p^{2ik}-1}{p^{i+1}}$  thus  $p^{n'} - a | p^n - 1$ . Hence,  $L$  splits over  $\mathbb{F}_{p^n}$ .

Moreover, using  $a = \frac{p^{i(2k-1)+1}}{p^{i+1}} = \sum_{j=0}^{2k-2} (-p^i)^j \leq p^{i(2k-1)}$ ,

we get  $\beta = \frac{\log_p(a) \cdot n}{n'^2} \leq \frac{i(2k-2) \cdot 2ik}{(i(2k-1))^2} = 1 - \frac{1}{(2k-1)^2} \leq 1$

2.  $r = \frac{p-k}{k-1} = \frac{k^n-1}{k-1}$  is clearly an integer since  $k-1 | k^n - 1$ . Also,  $r | p^n - 1$  since

$$\begin{aligned} p^n - 1 &= (k^n + k - 1)^n - 1 = \sum_{i=1}^n \binom{n}{i} (k^n - 1)^i k^{n-i} \\ &= r(k-1) \sum_{i=1}^n \binom{n}{i} (k^n - 1)^{i-1} k^{n-i} \end{aligned}$$

Moreover,  $p = \frac{p-k}{k-1}(k-1) + k = r(k-1) + k$  with  $k < 1 + k + \dots + k^{n-1} = r$  so  $a = (p \bmod r) = k$  and  $p - a = r(k-1)$

Therefore,  $p^n - 1 = (p-a) \sum_{i=1}^n \binom{n}{i} (k^n - 1)^{i-1} k^{n-i}$  and thus  $p-a | p^n - 1$ . So  $L = X^p - X^a$  splits in  $\mathbb{F}_{p^n}$ .

Furthermore, since  $k^n \leq k^n + k - 1 = p$ , we have

$$\beta = \log_p(a) \cdot n / 1 = \log_p(a^n) = \log_p(k^n) \leq 1$$

3. Let us first show that  $r = \frac{(p^n-1)(k-(-1)^n)}{(k^n-k)(k^n-(-1)^n)}$  is an integer ie that  $\frac{(k^n-k)(k^n-(-1)^n)}{k-(-1)^n} \in \mathbb{N}$  and divides  $p^n - 1$ .

For this we will show that  $\gcd((k^n - k), (k^n - (-1)^n)) = k - (-1)^n$ , that  $(k^n - k) | p^n - 1$  and that  $(k^n - (-1)^n) | p^n - 1$ .

First,  $\gcd((k^n - k), (k^n - (-1)^n)) = k - (-1)^n$  since  $k - (-1)^n$  divides both terms and  $(k^n - (-1)^n) - (k^n - k) = k - (-1)^n$ .

Moreover,  $(p^n - 1) = ((k^n - k) + (-1)^{n+1})^n - 1 = \sum_{i=1}^n \binom{n}{i} (k^n - k)^i (-1)^{(n+1)(n-i)}$  so  $(k^n - k) | p^n - 1$ .

Similarly,

$$\begin{aligned} (p^n - 1) &= ((k^n - (-1)^n) - k)^n - 1 \\ &= \sum_{i=1}^n \binom{n}{i} (k^n - (-1)^n)^i (-k)^{n-i} + (-k)^n - 1 \\ &= (k^n - (-1)^n) \left( \sum_{i=1}^n \binom{n}{i} (k^n - (-1)^n)^{i-1} (-k)^{n-i} + (-1)^n \right) \end{aligned}$$

so  $(k^n - (-1)^n) | p^n - 1$

Let us now split the study according to the parity of  $n$ .

- If  $n$  is even:

Let us show that  $(p^{n'} \bmod r) = p^{n'} - r \frac{k^n - (-1)^n}{k - (-1)^n}$

Indeed,  $c := p^{n'} - r \frac{k^n - (-1)^n}{k - (-1)^n} = p^{n-1} - \frac{p^n - 1}{k^n - k} = \frac{p^{n-1} + 1}{k^n - k} \simeq k^{n(n-1) - n} = k^{n^2 - 2n}$  while  $r \simeq k^{n^2 + 1 - 2n}$  for  $k^n \gg 1$ , so  $c/r \simeq 1/k$  and thus for  $k^n$  big enough  $c < r$ . So  $a := (p^{n'} \bmod r) = c$

Moreover  $p^{n'} - a = \frac{p^n - 1}{k^n - k} | p^n - 1$  so  $L$  splits over  $\mathbb{F}_{p^n}$

Furthermore, we see that showing that  $\beta = n \log_p(a)/(n-1)^2 \leq 1$  is equivalent to showing that  $a^n < p^{(n-1)^2}$ .

But  $\frac{a^n}{p^{(n-1)^2}} \simeq \frac{(k^{n^2 - 2n})^n}{k^{n(n-1)^2}} = (k^n)^{(n^2 - 2n - n^2 + 2n + 1)} = k^{-n} < 1$  when  $k^n \gg 1$  so  $\beta = n \log_p(a)/(n-1)^2 \leq 1$

- If  $n$  is odd:

Let us show that  $(p^{n'} \bmod r) = p^{n'} - r \frac{k^n - k}{k - (-1)^n}$ .

Indeed  $c := p^{n'} - r \frac{k^n - k}{k - (-1)^n} = p^{n-1} - \frac{p^n - 1}{k^n + 1} = \frac{p^{n-1}k + 1}{k^n + 1} \geq 0$

$$\begin{aligned}
\text{and } r &= \frac{(p^n - 1)(k + 1)}{(k^n - k)(k^n + 1)} \\
&= \frac{p^{n-1}}{k^n + 1} (p(k + 1)) \frac{1}{k^n(1 - k^{1-n})} - \frac{1}{(k^n - k)(k^n + 1)} \\
&= \frac{p^{n-1}}{k^n + 1} (k^{n+1} + k^n + o(k^3)) k^{-n} (1 + k^{1-n} + o(k^{1-n})) + o(1) \text{ for } k^n \gg 1 \\
&= \frac{p^{n-1}}{k^n + 1} (k + 1 + o(k^{3-n})) (1 + k^{1-n} + o(k^{1-n})) + o(1) \\
&= \frac{p^{n-1}}{k^n + 1} (k + 1 + o(k^{3-n})) + o(1) \\
&= \frac{p^{n-1}k + 1}{k^n + 1} - \frac{1}{k^n + 1} + \frac{p^{n-1}}{k^n + 1} (1 + o(k^{3-n})) + o(1) \\
&= c + \frac{p^{n-1}}{k^n + 1} (1 + o(k^{3-n})) + o(1) \\
&= c + \frac{p^{n-1}}{k^n + 1} (1 + o(1)) \text{ since } n \geq 3 \\
&> c
\end{aligned}$$

so  $c < r$  and thus  $a := (p^{n'} \bmod r) = c$

Moreover  $p^{n'} - a = \frac{p^n - 1}{k^n + 1} | p^n - 1$ .

The only remaining thing to prove is that  $\beta \leq 1$ . For this, we will as before show that  $a^n \leq p^{(n-1)^2}$ .

Indeed,

$$\begin{aligned}
a^n &= \left( \frac{p^{n-1}k + 1}{k^n + 1} \right)^n \\
&= \left( \frac{p^{n-1}k}{k^n + 1} + o(1) \right)^n = \frac{p^{(n-1)^2 + n - 1} k^n}{(k^n + 1)^n} + o\left( \frac{p^{n(n-1)} k^n}{k^{n^2}} \right) \\
&= p^{(n-1)^2} \frac{p^{n-1} k^n}{(k^n + 1)^n} + o(k^{n(n-1)^2}) \\
&= p^{(n-1)^2} \frac{(k^{n(n-1)} - (n-1)k^{1+n(n-2)} + o(nk^{1+n(n-2)})) k^n}{(1 + 1/k^n)^n} k^{-n^2} + o(k^{n(n-1)^2}) \\
&= p^{(n-1)^2} (1 - (n-1)k^{1-n} + o(nk^{1-n})) (1 - k^{-n} + o(k^{-n})) + o(k^{n(n-1)^2}) \\
&= p^{(n-1)^2} (1 - (n-1)k^{1-n} + o(nk^{1-n})) + o(k^{n(n-1)^2}) \\
&< p^{(n-1)^2}
\end{aligned}$$

□

An heuristic introduced in the appendix of [11] says that there are only rare parameters for which we can have a quasi-subfield multiplicative polynomials. It uses really similar arguments to the one introduced before about the case with  $n$  a Mersenne prime. The previous families show that this heuristic about the repartition of completely splitting polynomials in fact fails.

### 5.3 Critics of this definition of quasi-subfield polynomial

In this part, we admitted that we could use the same result about the complexity of the algorithm and thus keep the same definition about  $\beta$ . However, this cannot be done without more explanations. Indeed now  $|\mathcal{F}| \simeq |\mathcal{V}| = p^{n'} - a + 1$  which in some case can be very different from  $p^{n'}$ .

Therefore noting  $n'' = \log_p(p^{n'} - a + 1)$ , we should now consider only polynomials such that  $n.\ell/n''^2 \leq 1$ , which is a constraint more restrictive than the one used before.

However the families introduced before are still valid. Indeed for the first one,  $p^{n'} - a + 1 = \frac{p^{2ik} - 1}{p^i + 1} + 1 \simeq p^{i(2k-1)} = p^{n'}$  thus  $n'' \simeq n'$ . For the second one  $p^{n'} - a + 1 = p - a + 1 = (k^n + k - 1) - k + 1 = k^n$  is very close to  $p = p^{n'}$  if  $k$  or  $n$  is big enough. For the last one, in the even case  $p^{n'} - a + 1 = \frac{p^n - 1}{k^n - k} + 1 \simeq k^{n(n-1)} \simeq p^{n'}$  if  $k$  or  $n$  is big enough. In the odd case,  $p^{n'} - a + 1 = \frac{p^n - 1}{k^n + 1} + 1 \simeq k^{n(n-1)} \simeq p^{n'}$  if  $k$  or  $n$  is big enough.

## 6 Use of other algebraic groups

In Section 3 and Section 5, we considered additive and multiplicative groups to construct quasi-subfield polynomials. Therefore, we are also interested in knowing what would happen with other algebraic groups.

In *Galois invariant smoothness basis* [6], Couveignes and Lercier describe a way to extend a theory classically applied to additive and multiplicative groups to other commutative algebraic groups such that the torus and elliptic curves. We will try to explore their path in order to see how we could use these groups to deduce quasi-subfield polynomials.

### 6.1 Torus

In the part about the torus, Couveignes and Lercier consider the following setting:  $K = \mathbb{F}_p$  is a finite field not of characteristic two and  $D \in \mathbb{F}_p^*$  is not a square

in  $F_p$ . The group  $G(\mathbb{F}_p) = \{P = (U, V) \in \mathbb{P}^1(K), U^2 - DV^2 \neq 0\}$  has order  $p + 1$  and the affine coordinates  $u(P) = U/V$  lie in  $\mathbb{F}_p \cup \{\infty\}$ . We call  $G(\mathbb{F}_p)$  a torus. The unit element is  $0_G = (1, 0)$ . The addition law on the group is defined as follow for elements  $P_1$  and  $P_2$  which are not the unit:  $u(P_1 \oplus_G P_2) = \frac{u(P_1)u(P_2)+D}{u(P_1)+u(P_2)}$  and  $u(\ominus_G(P_1)) = -u(P_1)$ . Let  $n \geq 2$  which divides  $p + 1$  and  $a$  a generator of  $G(\mathbb{F}_p)$ . Let  $I$  be the multiplication by  $n$  isogeny.

Then  $P_a(X) = \prod_{b \in I^{-1}(a)} (X - u(b))$  is irreducible in  $F_p[X]$  but completely splits over  $K = \mathbb{F}_{p^n}$ . An explicit description of  $P_a$  is

$$P_a(X) = \sum_{0 \leq 2k \leq n} X^{n-2k} \binom{n}{2k} D^k - u(a) \sum_{1 \leq 2k+1 \leq n} X^{n-2k-1} \binom{n}{2k+1} D^k$$

How can we deduce from  $P_a(X)$  a quasi-subfield polynomial? It is not obvious since the degree  $P_a$  is not a power of  $p$  and  $P_a$  is not sparse.

By construction, if  $a \neq a'$  then the roots to  $P_a$  and  $P_{a'}$  are distinct so  $P_a P_{a'}$  also completely splits over  $\mathbb{F}_{p^d}$ . Therefore we may look for quasi-subfield polynomials among the products of different  $P_a$ . However  $\deg P_a = n|p + 1$  thus  $\deg P_a$  does not divide  $p$  and thus there is no chance of finding a polynomial of degree a power of  $p$  by merely multiplying a few  $P_a$ . Nonetheless, as in the case of multiplicative subgroups, one can cope with this problem by multiplying by  $X$  until reaching a degree which is a power of  $p$ .

We did not find any quasi-subfield polynomials whose roots form a torus. However, this goal is maybe accessible.

For example, the product of  $P_a$  and  $P_{\ominus a}$  is sparse: half of its coefficients are zeroes.

$$\begin{aligned} \text{Indeed, } P_a \cdot P_{\ominus a} &= \prod_{b \in I^{-1}(a)} (X - u(b)) \prod_{b \in I^{-1}(\ominus a)} (X - u(b)) \\ &= \prod_{b \in I^{-1}(a)} (X - u(b)) \cdot \prod_{b \in I^{-1}(a)} (X - u(\ominus b)) \\ &= \prod_{b \in I^{-1}(a)} (X - u(b))(X + u(b)) \\ &= \prod_{b \in I^{-1}(a)} (X^2 - u(b)^2) \end{aligned}$$

which has only monomials of even degree.

Therefore, we may hope to reach sparse enough polynomials which would be good candidates for quasi-subfield polynomials.

## 6.2 Elliptic curves

The main problem while studying the torus is that it imposes the condition  $n|p + 1$ . To get rid of this condition, Couveignes and Lercier extend their ideas to elliptic curves and succeed to find a solution to their problem for any  $(p, n)$  such that

- $n$  is odd
- $n < (\sqrt{p} + 1)^2$
- there is a square-free multiple  $D$  of  $n$  such that  $D \not\equiv 1 \pmod{p}$  and

$$(\sqrt{p} + 1)^2 < D < (\sqrt{p} + 1)^2$$

Since their problem is really different from ours, we cannot be sure that this approach will be as successful in our case. Nonetheless, the conditions on  $n$  and  $p$  are still restrictive as they force  $n$  to be of the same order of magnitude that  $p$  or smaller than  $p$ . In particular, the case of the field  $F_{2^n}$  with  $n$  a big prime cannot be dealt with this approach.



## 7 Conclusion

In this thesis, we focused on building better families of quasi-subfield polynomials than the one introduced in [11]. We succeeded to find five new families based on additive and multiplicative groups. They lead to a more efficient ECDLP algorithm than the exhaustive search for wider families of  $p$  and  $n$  than what the original paper provided. For the specific case of *linearized* quasi-subfield polynomials, we ruled out the existence of quasi-subfield polynomials where  $\deg \lambda$  is small enough to improve on the generic algorithms for ECDLP (or it would require a major breakthrough in the estimation of the complexity of the algorithm used here).

We mainly studied additive and multiplicative groups. Further research may study other families of multiplicative quasi-subfield polynomials, in particular when  $p$  is a Solinas prime. Another interesting direction is the use of other algebraic groups, as suggested in the last section, in order to find quasi-subfield polynomials with much smaller  $\beta$ . Lastly, the question of the existence of non-linearized quasi-subfield polynomials where  $\deg \lambda$  is small enough to improve on generic algorithms remains open. It is linked with the question of the existence of a numerical lower bound on  $\beta$ , similar to the one given by Theorem 1.2, but valid for any quasi-subfield polynomial. An interesting candidate would be 0.1. as we found no quasi-subfield polynomial with  $\beta$  less than 0.1 and that it is the bound determining whether the quasi-subfield polynomials can bring a significant impact to the resolution of the ECDLP.

## References

- [1] The on-line encyclopedia of integer sequences (OEIS). <https://oeis.org/>.
- [2] SageMath - open-source mathematical software system. <http://www.sagemath.org/>.
- [3] Elwyn Berlekamp. *Algebraic coding theory*. World Scientific, 1968.
- [4] William Y. C. Chen and James D. Louck. The combinatorial power of the companion matrix. *Linear Algebra and its Applications*, 232:261–278, 1996.
- [5] Information Technology Laboratory Computer Security Division. Transition plans for key establishment schemes | CSRC. <https://csrc.nist.gov/News/2017/Transition-Plans-for-Key-Establishment-Schemes>.

- [6] Jean-Marc Couveignes and Reynald Lercier. Galois invariant smoothness basis. *arXiv:0802.0282 [math]*, 2008.
- [7] Claus Diem. On the discrete logarithm problem in elliptic curves. *Compositio Mathematica*, 147(1):75–104.
- [8] Steven D. Galbraith and Pierrick Gaudry. Recent progress on the elliptic curve discrete logarithm problem. *Designs, Codes and Cryptography*, 78(1):51–72, 01 2016.
- [9] Pierrick Gaudry. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *Journal of Symbolic Computation*, 44(12):1690–1702, 2009.
- [10] Mike Hamburg. Ed448-goldilocks, a new elliptic curve. <https://eprint.iacr.org/2015/625>.
- [11] Ming-Deh Huang, Michiel Kusters, Christophe Petit, Sze Ling Yeo, and Yang Yun. Quasi-subfield polynomials and the elliptic curve discrete logarithm problem. *Journal of Mathematical Cryptology*, 11 2018.
- [12] Gary McGuire and Daniela Mueller. Some results on linearized trinomials that split completely. 2019.
- [13] Gary McGuire and John Sheekey. A characterization of the number of roots of linearized and projective polynomials in the field of coefficients. *Finite Fields and Their Applications*, 57:68–91, 2019.
- [14] NIST. Recommended elliptic curves for federal government use, 1999.
- [15] J. Maurice Rojas. Solving degenerate sparse polynomial systems faster. *Journal of Symbolic Computation*, 28(1):155–186, 1999.
- [16] Igor Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. <http://eprint.iacr.org/2004/031>.
- [17] Jerome A Solinas et al. *Generalized mersenne numbers*. 1999.
- [18] Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptology*, 12:1–28, 1999.
- [19] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32(1):54–62, 1986.

# A Appendix

## A.1 Omitted proofs

### A.1.1 Proof of Lemma 1.3

Let us first prove the condition about the exhaustive search. In order to beat exhaustive search, we know that we need  $m > 2$ .

Therefore,  $1 - \alpha + \frac{\alpha}{m} + c_{algo}\beta\alpha^2 \leq 1 - \frac{2\alpha}{3} + c_{algo}\beta\alpha^2$ . Moreover,  $1 - \frac{2\alpha}{3}\beta\alpha^2 = 1$  if and only if  $\alpha = 0$  or  $-\frac{2}{3} + c_{algo}\beta\alpha = 0$  ie  $\alpha = \frac{2/3}{c_{algo}\beta} = 4/3\alpha_\beta$ . Hence, for all  $\alpha \in ]0, 4/3\alpha_\beta[$ ,  $1 - \alpha + \frac{\alpha}{m} + c_{algo}\beta\alpha^2 < 1$ .

Further more, if  $4/3\alpha_\beta \frac{n}{n'} \geq 3$ , then  $\mathbb{Z}_{>2} \cap \{\alpha \frac{n}{n'}, 0 < \alpha < 4/3\alpha_\beta\} \neq \emptyset$ . Thus we can choose an integer  $m$  in this set which will allow us to get a better complexity than the brute-force algorithms.

$$\text{Finally, } 4/3\alpha_\beta \frac{n}{n'} \geq 3 \Leftrightarrow \alpha_\beta \geq \frac{9n'}{4n}$$

**Remark A.1.** If we just considered that  $1/m$  was negligible, we would only need to know when  $-\alpha + c_{algo}\beta\alpha^2 = 0$  which happens when  $\alpha = 0$  or  $\alpha = \frac{1}{c_{algo}\beta} = 2\alpha_\beta$ . Therefore to find  $m \geq 3$  in  $[0, 2\alpha_\beta \frac{n}{n'}]$ , we would have needed  $\alpha_\beta > \frac{3n'}{2n}$ . Moreover, to keep the assumption  $m \gg 1$  we would need  $2\alpha_\beta \frac{n}{n'} \gg 1$ . This leads to the condition  $\alpha_\beta \gg \frac{n'}{2n}$  which is coherent with the previously given bound.

Let us now look at the requirements needed to beat the generic algorithms.

Here we only consider the case  $m$  big enough to consider  $1/m$  as negligible. This time we need to find when  $1 - \alpha + c_{algo}\beta\alpha^2 < 1/2$ . Therefore we need to search the roots of  $1/2 - \alpha + c_{algo}\beta\alpha^2$ . The discriminant is  $\Delta = 1 - \frac{4c_{algo}\beta}{2} = 1 - \frac{1}{\alpha_\beta}$ . It is non-negative if and only  $\alpha_\beta \geq 1$  and the roots are  $\alpha_\pm = \frac{1 \pm \sqrt{\Delta}}{2c_{algo}\beta} = \left(1 \pm \sqrt{1 - \frac{1}{\alpha_\beta}}\right) \alpha_\beta$ . Hence, for all  $\alpha \in ]\alpha_-, \alpha_+[$ ,  $1 - \alpha + c_{algo}\beta\alpha^2 < 1/2$ .

Further more, if  $\frac{\alpha_+ n}{n'} \gg 1$  and  $\alpha_+ \frac{n}{n'} > \alpha_- \frac{n}{n'} + 1$ , then  $\mathbb{Z}_{\gg 1} \cap \{\alpha \frac{n}{n'}, \alpha_- < \alpha < \alpha_+\} \neq \emptyset$ . Thus we can choose an integer  $m$  in this set which will allow us to get a better complexity than the generic algorithms.

Finally,

$$\begin{aligned}
\frac{\alpha_+ n}{n'} \gg 1 &\Leftrightarrow \left(1 + \sqrt{1 - \frac{1}{\alpha_\beta}}\right) \alpha_\beta \frac{n}{n'} \gg 1 \\
&\Leftrightarrow \sqrt{1 - \frac{1}{\alpha_\beta}} \gg \frac{n'}{n\alpha_\beta} - 1 \\
&\Leftrightarrow 1 - \frac{1}{\alpha_\beta} \gg \left(\frac{n'}{n\alpha_\beta}\right)^2 - 2 \cdot \frac{n'}{n\alpha_\beta} + 1 \\
&\Leftrightarrow 0 \gg \frac{n'^2}{n^2\alpha_\beta} - 2\frac{n'}{n} + 1 \\
&\Leftrightarrow 0 \gg \frac{n'^2}{n^2} - 2\alpha_\beta \frac{n'}{n} + \alpha_\beta, \quad \Delta = 4(\alpha_\beta^2 - \alpha_\beta) > 0 \text{ if } \alpha_\beta > 1 \\
&\Leftrightarrow \left|\frac{n'}{n} - \alpha_\beta\right| \gg \sqrt{\Delta}/2 = \sqrt{\alpha_\beta^2 - \alpha_\beta}
\end{aligned}$$

and

$$\begin{aligned}
\alpha_+ \frac{n}{n'} > \alpha_- \frac{n}{n'} + 1 &\Leftrightarrow (\alpha_+ - \alpha_-) \frac{n}{n'} > 1 \\
&\Leftrightarrow 2\sqrt{1 - \frac{1}{\alpha_\beta}} \alpha_\beta \frac{n}{n'} > 1 \\
&\Leftrightarrow 4\left(1 - \frac{1}{\alpha_\beta}\right) \alpha_\beta^2 \frac{n^2}{n'^2} > 1 \\
&\Leftrightarrow \alpha_\beta^2 - \alpha_\beta > \frac{n'^2}{4n^2} \\
&\Leftrightarrow \alpha_\beta > \frac{1 + \sqrt{1 + n'^2/n^2}}{2}
\end{aligned}$$

As it was forecast, this condition is stronger than simply  $\alpha_\beta > 1$  as it is in the ideal case. Moreover, it imposes  $n'/n$  small compared to  $\alpha_\beta$  so  $|\frac{n'}{n} - \alpha_\beta| \gg \sqrt{\alpha_\beta^2 - \alpha_\beta}$  becomes  $\frac{n'}{n} \ll \alpha_\beta - \sqrt{\alpha_\beta^2 - \alpha_\beta}$

### A.1.2 The formula for the power of companion matrices

In [4], the main result is a formula to give the value of all the coefficients in the power of companion matrices. It is a result based on a combinatorial analysis of a digraph.

$$\text{It says that for any companion matrix } M = \begin{bmatrix} u_1 & u_2 & \dots & 0 & u_m \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix},$$

and any  $r \geq 1$ , we have

$$M_{i,j}^r = \begin{cases} 1 & \text{if } r = i - j \\ \sum_{k_1, \dots, k_m} w_{\mathbf{k}} u_1^{k_1} \dots u_m^{k_m} & \text{otherwise} \end{cases}$$

where  $\mathbf{k} = (k_i)_{1 \leq i \leq m}$  are non-negative integers such that  $k_1 + 2k_2 + \dots + mk_m = r - i + j$  and  $w_{\mathbf{k}} = \frac{k_j + \dots + k_m}{k_1 + \dots + k_m} \binom{k_1 + \dots + k_m}{k_1, \dots, k_m}$ .

This result is really useful for us as we need to compute the powers of a companion matrix.

However in our case, the shape of the matrix is  $N = \begin{bmatrix} 0 & 0 & \dots & 0 & u_m \\ 1 & 0 & \dots & 0 & u_{m-1} \\ 0 & 1 & \dots & 0 & \vdots \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & 1 & u_1 \end{bmatrix}$ .

Thus  $N = f(M)$  with  $f$  the involution defined by

$$f : (a_{i,j})_{1 \leq i,j \leq m} \mapsto (a_{m-j+1}, b_{m-i+1})_{1 \leq i,j \leq m}$$

which sends square matrices of size  $m$  on square matrices of size  $m$ .

Let us show that for any A and B square matrices of size  $m$ ,  $f(AB) = f(B)f(A)$ , in order to have  $N^r = f(M)^r = f(M^r)$  and thus deduce the coefficients of  $N^r$  for any  $r \geq 1$ . We define  $C = AB$  and  $D = f(AB)$ ,  $E = f(A)$ ,  $F = f(B)$  and  $G = f(B)f(A)$ .

Then  $c_{i,j} = \sum_{k=1}^m a_{i,k}b_{k,j}$ , so

$$\begin{aligned} d_{i,j} = c_{m-j+1,m-i+1} &= \sum_{k=1}^m a_{m-j+1,k}b_{k,m-i+1} \\ &= \sum_{k=1}^m e_{m-k+1,j}f_{i,m-k+1} \\ &= \sum_{k=1}^m f_{i,k}e_{k,j} \\ &= g_{i,j} \end{aligned}$$

Hence  $N^r = f(M^r)$  so  $N_{i,j}^r = M_{m-j+1,m-i+1}^r$

$$N_{i,j}^r = \begin{cases} 1 & \text{if } r = (m-j+1) - (m-i+1) = i-j \\ \sum_{\mathbf{k}} w_{\mathbf{k}} u_1^{k_1} \dots u_m^{k_m} & \text{otherwise} \end{cases}$$

where  $\mathbf{k} = (k_i)_{1 \leq i \leq m}$  are non-negative integers such that

$$k_1 + 2k_2 + \dots + mk_m = r - (m-j+1) + (m-i+1) = r - i + j$$

and  $w_{\mathbf{k}} = \frac{k_{m-i+1} + \dots + k_m}{k_1 + \dots + k_m} \binom{k_1 + \dots + k_m}{k_1, \dots, k_m}$ , which is the result that we use in 4.3.

## A.2 Systematic search of quasi-subfield polynomials

### A.2.1 Search of linearized polynomials

Let us now present the code written with SageMath to search linearized quasisubfield polynomials.

The class *linearized\_poly* represents a linearized polynomial and possesses all the functions needed to decide whether it is a quasi-subfield polynomial, for which  $n$ , and if it belongs to one of the known families of quasi-subfield polynomials.

It is used inside a function *search* which explores all the possible coefficients.

```

class linearized_poly:
    """ a linearized polynomial """

    def __init__(self, matrix, p):
        """ initialised by giving the companion matrix  $\mathcal{E}$ 
            the characteristic of the field  $p$  """
        self.matrix=matrix # C_L
        self.coefs=[-c for c in matrix[-1]]+[Zmod(p)(1)] # L
            = sum_i(coefs[i]X^(p^i))
        self.n_dash = matrix.nrows() # n'
        self.l= self.l_index() # l
        self.p=p # p
        self.simple=self.coefs

    def l_index(self):
        """ return the value of  $l$  """
        for i in range(len(self.coefs)-2,0,-1):
            if self.coefs[i]!=0:
                return i
        return 0

    def f(self):
        """return  $f$  such that  $L_f$  is the polynomial
            considered """
        poly=0
        x=PolynomialRing(Zmod(self.p), 'X').gen()
        for i in range(len(self.simple)):
            c=self.simple[i]
            poly+=c*x^(i)
        return poly

    def quasi_subfield(self, max_b=1):
        """ return True iff it is a max_b-quasi-subfield
            polynomial and the gcd of the indices of the
            coefficients and of  $n$  is 1
            In this case, initialise self.n to the first
            value of  $n$  where  $C_L^n=I$  """
        if self.l>0:
            M=self.matrix
            for n in range(2, int(max_b*self.n_dash^2/self.l)
                +1):

```

```

M=M*self.matrix
if n>self.n_dash and M.is_one():
    self.beta=(self.l*n*1.0)/self.n_dash**2
    self.n=n
    J=[self.n]
    for i in range(len(self.coeffs)):
        if self.coeffs[i]!=0:
            J.append(i)
    return gcd(J)==1
return False

def to_string(self):
    return " ".join([str(c) for c in self.coeffs])

def to_string_latex(self):
    s=""
    for i in range(len(self.simple)-1,0,-1):
        if self.simple[i]!=0:
            if self.simple[i]==Zmod(self.p)(1):
                s+="X^{ "+str(i)+" }"
            elif self.simple[i]==Zmod(self.p)(-1):
                s+="-X^{ "+str(i)+" }"
            else:
                s+=" "+str(self.simple[i])+"X^{ "+str(i)+
                " }"
    s+=" "+str(self.simple[0])
    return s[1:]

def equivalent_class(self):
    """ Once we know that it is a quasi-subfield
        polynomial, we compute all the elements of its
        equivalent class which share the same n """
    set_class =set([self.to_string()])
    L=[]

    for a in Zmod(self.p):
        if a^self.n==1 and a!=1:
            coeff_3 = [self.coeffs[i] *a**(i-self.n_dash
            ) for i in range(len(self.coeffs))]
            L.append(coeff_3)
            set_class.add(" ".join([str(c) for c in
            coeff_3]))
        if (self.p<=2 or GF(self.p)(2) not in
            coeff_3) and (self.p<=3 or GF(self.p)(3)
            not in coeff_3):
            self.simple=coeff_3

```

```

### not useful when considering only coefficients
    in Fp since the frobenius leaves them unchanged
#for b in Zmod(self.p):
#    if b!=0 and b!=1:
#        for new_co in L:
#            coeff_4 = [new_co[i]*b^(self.p^i-self.p
#            ^self.n_dash) for i in range(len(new_co))]
#            set_class.add(" ".join([str(c) for c
#            in coeff_4]))
#            if (self.p<=2 or GF(self.p)(2) not in
#            coeff_4) and (self.p<=3 or GF(self.p)(3) not in
#            coeff_4):
#                self.simple=coeff_4

```

```

return set_class

```

```

def analyse_shape(self, set_p):
    """ Try to recognize if the quasi-subfield belongs
        to one of the families we identified """
    list_i = [i for i in range(len(self.simple)) if self
        .simple[i]!=0]
    type_1, type_2, type_3= True, True, True
    #pre-analysis:
    for i in list_i[1:]:
        if self.simple[i]!=1:
            type_0, type_1=False, False

    type_1 = type_1 and (list_i[0]==0) and (self.simple[
        list_i[0]]==1)
    type_2 = type_2 and ((list_i[0]==0 and self.simple[
        list_i[0]]==1) or list_i[0]==1)

    # type 1:  $X+X^{p^{p_0}}+\dots + X^{p^{p_d}}$ , where
     $q'=p^r$ ,  $n=p-\{d+1\}$ ,  $p-a=1+q'+\dots+q'^a = (q'^{a+1}-1)/(q'-1)$ 
    if list_i !=[0,1]:
        qq=list_i[2]-1
        type_1= type_1 and (int(log(qq, self.p))==log(qq,
            self.p))
        a=0
        p_a=1
        while p_a <= list_i[-1]:
            if a+1>=len(list_i) or list_i[a+1]!=p_a:
                type_0 =False
            a+=1

```



```

        p_a=qq*p_a+1

# type 2:  $aX+X^p+X^{p^q}+\dots+X^{p^{q^d}}$  or  $X+X^{p^{q-1}}+\dots+X^{p^{q^d-1}}$ ,  $n = (q^{d+1}-1)$ 
if list_i[1]==1:
    q=list_i[2]
    #a!=0:
    type_2_a=(int(log(q, self.p))==log(q, self.p)) and
        (list_i == [0]+[q^i for i in range(int(log(
            len(self.simple), q)+1)])
    if self.p!=2:
        type_2=type_2 and type_2_a
    else:
        q=list_i[1]+1
        type_2_0 = (int(log(q, self.p))==log(q, self.p)
            ) and (list_i == [0]+[q^i-1 for i in
                range(1, int(log(len(self.simple), q)+1)])
        type_2=type_2 and (type_2_a or type_2_0)
else :
    q=list_i[1]+1
    type_2 = (int(log(q, self.p))==log(q, self.p)) and
        (list_i == [0]+[q^i-1 for i in range(1, int(
            log(len(self.simple), q)+1)])

# type 3: inverses of type 1 and 2
x=PolynomialRing(Zmod(self.p), 'X').gen()
inverse =(x^self.n-1)//self.f()
type_3 = "_" .join([str(c) for c in inverse.
    coefficients(sparse=False)]) in set_p
if type_3:
    inverse = " _ _ _ (" + str(inverse) + ")"
else:
    inverse = ""
return ("&" .join(["X" if c else "_" for c in [type_1
    , type_2 , type_3 ]]) + inverse + "\\ \\ \\")

def search(p, max_n_dash=40):
    """output all the quasi-subfield polynomials in  $F_p[X]$ 
    of degree less than X and print representative of
    each equivalence class encountered"""
    set_p=set() # the set of the known quasi-
        subfield polynomials
    Zp=Zmod(p)
    set_c=set([Zp(0), Zp(1), Zp(-1)]) # or set(Zp) depending
        of the value of n_dash we want to reach

```

```

stack = [[]] # where we accumulate the
             # coefficients in order to generate all the (a_0, a_1, \
             # dots , a_(n'-1)) possible
n_dash = -1
while stack != []:
    coeffs = stack.pop(0)
    if len(coeffs) + 1 > n_dash:
        n_dash = len(coeffs) + 1
        M = Matrix.companion([Zp(0) for k in range(n_dash)
                               ] + [Zp(1)]) . transpose()
        if n_dash > max_n_dash:
            break
    if n_dash > 0:
        for c in set_c:
            co = [c] + coeffs
            if c != 0:
                M[n_dash - 1] = co
                P = linearized_poly(M, p)
                if P.to_string() not in set_p and P.
                   quasi_subfield(max_b=1):
                    for poly in P.equivalent_class():
                        set_p.add(poly)
                    print("&" . join([str(c) for c in [P.
                                                       to_string_latex(), P.n, str(P.beta)
                                                       [:3], p, P.analyse_shape(set_p) ]]))
            stack.append(co)
return set_p

```

For example `search(2, max_n_dash=16)` will output all the quasi-subfield polynomials in  $\mathbb{F}_2[X]$  of degree less than 16.

### A.2.2 Search of multiplicative quasi-subfield polynomials

We can also use Sage to search multiplicative quasi-subfield polynomials. To do this, we search tuples of  $(p, n, n', r)$  verify  $r|p^{n'} - a|p^n - 1$  with  $a = p^{n'} \pmod r$  and  $\beta = \frac{n \cdot \log_p(a)}{n'^2} \leq 1$ . Let us show here the results for small  $n$ . We increase  $p$  as long as to find a few QSP per  $n$ . When two values of  $r$  for the same  $(p, n, n')$  can be used and leads to the the same QSP, we only write one set of working parameters.

The code in Sage is the following:

```

print("n&p&nn&r &a &beta &_1&_2&_3_\\")
for n in range(2, 25): #We try all the small n
    p=2
    set_poly=set() # We keep track of the
                  # polynomials known for this n in order to avoid

```

```

printing them several time.
while p<1000 and len(set_poly)<10: # We try all the
small p
    if is_prime(p):
        for r in divisors(p^n-1):
            for nn in range(1,n): # nn =n'
                a=p^nn %r
                beta = float(log(a,p)*n/(nn^2))
                poly = "X^"+str(p^nn)+"_X^"+str(a)
                if (not poly in set_poly) and a!=p^nn
                    and (p^n-1)%(p^nn-a)==0 and beta <=1
                    and beta >0:
                        print(str(n)+"&"+str(p)+"&"+str(nn)+
                            "&"+sci(r)+"&"+sci(a)+"&"+str(
                                beta)[:4]+"&"+get_type(p,n,nn,r,a)
                                )+"\\\\"")
                        set_poly.add(poly)
p=next_prime(p)

```

We use here a function *get\_type* which we implemented to recognize the families listed in Proposition 5.1.

Let us display here some of the output of this code. We to cut it to present only the most significant lines but a thing important to keep in mind is that we were not able to classify all the output in families. Surely, much more families remains to be conjectured and proved.

$n$	$p$	$n'$	$r$	$a$	$\beta$	1	2	3
2	5	1	3	2	0.86		X	
2	11	1	4	3	0.91		X	
2	19	1	5	4	0.94		X	
2	29	1	6	5	0.95		X	
2	41	1	7	6	0.96		X	
3	7	2	19	11	0.92			X
3	29	1	13	3	0.97		X	
3	61	2	291	229	0.99			X
3	67	1	21	4	0.98		X	
3	211	2	1443	1231	0.99			X
4	2	3	5	3	0.70	X		
4	3	3	10	7	0.78	X		
4	5	3	26	21	0.84	X		
4	7	3	50	43	0.85	X		

$n$	$p$	$n'$	$r$	$a$	$\beta$	1	2	3
4	11	3	122	111	0.87	X		
4	3	3	16	11	0.97			
			$\vdots$					
4	17	1	15	2	0.97		X	
4	83	1	40	3	0.99		X	
5	3	3	11	5	0.81			
5	5	2	11	3	0.85			
5	31	4	86755	55971	0.99			X
5	37	2	33	16	0.95			
5	109	2	62	39	0.97			
5	241	4	$5.5 * 10^7$	$4.1 * 10^7$	0.99			X
5	307	2	5231	91	0.98			
5	1021	4	$5.3 * 10^9$	$4.2 * 10^9$	0.99			X
5	3121	4	$1.8 * 10^{11}$	$1.5 * 10^{11}$	0.99			X
6	2	5	21	11	0.83	X		
6	3	2	7	2	0.94			
6	3	5	91	61	0.89	X		
6	5	4	93	67	0.97			
6	5	5	651	521	0.93	X		
6	7	5	2451	2101	0.94	X		
6	11	3	37	36	0.99			
6	11	4	703	581	0.99			
			$\vdots$					
6	4099	1	1365	4	0.99		X	
7	127	6	$9.8 * 10^{10}$	$6.5 * 10^{10}$	0.99			X
7	16381	6	$5.8 * 10^{21}$	$4.7 * 10^{21}$	0.99			X
7	78121	6	$1.7 * 10^{25}$	$1.4 * 10^{25}$	0.99			X
8	2	6	17	13	0.82	X		
8	2	7	85	43	0.88	X		
8	3	6	82	73	0.86	X		
8	3	7	820	547	0.93	X		
8	5	6	626	601	0.88	X		
9	19681	8	$4.5 * 10^{30}$	$3.4 * 10^{30}$	0.99			X
9	262147	1	87381	4	0.99		X	

$n$	$p$	$n'$	$r$	$a$	$\beta$	1	2	3
10	2	9	341	171	0.91	X		
10	3	9	7381	4921	0.95	X		
10	5	9	$4.0 * 10^5$	$3.2 * 10^5$	0.97	X		
10	7	9	$5.8 * 10^6$	$5.0 * 10^6$	0.97	X		
10	11	9	$2.1 * 10^8$	$1.9 * 10^8$	0.98	X		
11	4194301	10	$2.0 * 10^{60}$	$1.6 * 10^{60}$	0.99			X
12	2	7	13	11	0.84			
12	2	9	65	57	0.86	X		
12	2	10	273	205	0.92	X		
12	2	11	1365	683	0.93	X		
12	3	9	730	703	0.88	X		
12	3	10	6643	5905	0.94	X		
12	3	11	66430	44287	0.96	X		

Some multiplicative quasi-subfield polynomials output by our Sage function