On some algorithms related to matrices with coefficients in a finite field and their computational complexity

DOI: https://doi.org/10.54654/isj.v1i21.1032

Pablo Freyre Arrozarena, Alejandro Freyre Echevarría, Ramses Rodríguez Aulet, Ernesto Domínguez Fiallo, Samir Alzugaray Vizcaino

***Abstract*— In the literature survey, there exist several research papers devoted to the generation of non-singular matrices with coefficients in a finite field. One of these papers refers to the generation of such matrices through the multiplication of polynomials modulus a primitive polynomial. However, the complexity bound given for the algorithm is not accurate. Thus, in this paper, we conduct a new analysis on the complexity of the same. We also remove the restriction of using a primitive polynomial to generate the matrix by using an arbitrary monic polynomial over a finite field whose independent term is distinct from zero.**

***Tóm tắt*— Qua nghiên cứu thấy rằng, tồn tại nhiều phương pháp sinh các ma trận không suy biến trên trường hữu hạn. Một trong những công bố khoa học đã đề cập đến việc tạo ra các ma trận như vậy thông qua phép nhân của các đa thức theo mô-đun một đa thức nguyên thủy. Tuy nhiên, đánh giá độ phức tạp của thuật toán được đưa ra là chưa chính xác. Do đó, trong bài báo này, chúng tôi đề xuất một phân tích mới cho độ phức tạp để khắc phục hạn chế trong các công bố trước. Nó liên quan đến việc sử dụng đa thức nguyên thủy để tạo ma trận bằng cách sử dụng đa thức monic tùy ý trên một trường hữu hạn có số hạng độc lập khác 0.**

***Keywords*—non-singular matrices, multiplication of polynomials, computational complexity.**

***Từ khóa*— ma trận không suy biến, phép nhân đa thức, độ phức tạp tính toán.**

I. INTRODUCTION

The problems related to the generation of non-singular matrices, find their inverses and multiply two matrices are classical in linear algebra. All these problems have complexity $\mathcal{O}(n^3)$ when solved using naive algorithms [1]. However, there exist methods whose computational complexity is lower than this bound, such as the one presented in [2] for matrix multiplication which runs in $\mathcal{O}(n^{2.37286})$. For the case of generating non-singular matrices, it is known that on average it takes between 3 and 4 random selections to obtain a non-singular matrix over the finite field $\mathbb{F}_2$, which consumes about $3n^2$ or $4n^2$ bits (n^2 bits each time) [2, 3]. In addition, the work from Randall [4] shows an efficient algorithm for the generation of a random non-singular matrix over $\mathbb{F}_q$. In the particular case of $\mathbb{F}_q = \mathbb{F}_2$, the algorithm has a time complexity of $M(n) + \mathcal{O}(n^2)$ where $M(n)$ is the computational complexity for multiplying two $n \times n$ matrices. Finally, one can generate non-singular matrices through the multiplication of polynomials modulus a primitive polynomial over a finite field as presented in [5].

When working in the field $\mathbb{F}_q$ such that $q = p^t$ with p a prime number and $t \in \mathbb{Z}^+$, the latter method is stated to have computational complexity $\mathcal{O}(n^3(\log_2 n)(\log_2 \log_2 n))$ [5], which we believe is not an accurate upper bound for the algorithm. Moreover, as the size of the matrix grows, it will be difficult to execute the algorithm due the complexity of finding primitive polynomials with a high degree.

Our contributions: This paper addresses the complexity of the algorithms presented in [5],

This manuscript is received on May 18, 2024. It is commented on June 4, 2024 and is accepted on June 27, 2024 by the first reviewer. It is commented on June 5, 2024 and is accepted on June 5, 2024 by the second reviewer.

showing that the upper bound presented by its authors is not accurate and providing the mathematical proofs that allow us to reduce the upper bound of such algorithms, also removing the constraint of using primitive polynomials as a mandatory input parameter by substituting them with a monic polynomial with a non-zero independent coefficient. Furthermore, we analyze special cases of the input parameters of the algorithms described later in this paper that allow us to reduce their computational complexity to be close to those in the state-of-the-art of non-singular matrix generation. Finally, we remark that the matrices obtained through our algorithms can be used in key-dependent encryption schemes where part of our algorithm's input parameters are derived from the key. The output of our algorithms can also be used for McEliece-type cryptosystems.

II. Mathematical Preliminaries

Let $\mathbb{F}_q$ denote the finite field q elements, and G be an arbitrary group acting over an arbitrary set Ω. We denote by $\beta^g \in \Omega$ the action of $g \in G$ on $\beta \in \Omega$. The stabilizer of points $\beta_1, \beta_2, \dots, \beta_{i-1} \in \Omega$ in G is denoted by $G^{(i)} = G_{\beta_1,\beta_2,\dots,\beta_{i-1}}, G = G^{(1)}$ *and* $G^{(k+1)} = I_n$. Let $B = (\beta_1, \beta_2, \dots, \beta_k)$ be a basis for G, then the basic orbits are the sets of points $\Delta^{(i)} = \left\{\beta_i^{G^{(i)}}, i = 1,2, \dots, k\right\}$.

For a given base B, the Schreier structure [6] is defined as the arrangement $L = [L\beta_1, L\beta_2, \dots, L\beta_k]$ in which

$$L\beta_i = \begin{bmatrix} \beta_i, & \alpha_1, & \alpha_2, \dots, & \alpha_{s_i} \\ I_n, & g_1^{(i)}, & g_2^{(i)}, \dots, & g_{s_i}^{(i)} \end{bmatrix}$$

where:

- $\Delta^{(i)} = (\beta_i, \alpha_1, \alpha_2, \dots, \alpha_{s_i})$
- $g_1^{(i)}, g_2^{(i)}, \dots, g_{s_i}^{(i)} \in G^{(i)}$
- $\beta_i^{g_1^{(i)}\, g_2^{(i)} \dots g_j^{(i)}} = \alpha_j, i = 1, 2, \dots, k$ *and* $j = 1, 2, \dots, s_i$.

We call a right transversal for $G^{(i+1)}$ in $G^{(i)}$ to the set $U_i = \{x_0, x_1, \dots, x_{s_i}\}$ of the representatives of the right cosets of $G^{(i+1)}$ in $G^{(i)}$, being $x_0 = 1_G$ and $s_i + 1$ the index of $G^{(i+1)}$ in $G^{(i)}$. Every element $g \in G$ can be expressed in a unique way as the product of elements of a right transversal $U_i,\ i = 1, 2, \dots, k$ for $G^{(i+1)}$ in $G^{(i)}$, i.e., $\forall g \in G, g = \prod_{i=k}^{1} u_i$ where $u_i \in U_i = \left\{\prod_{j=0}^{l} g_j^{(i)} : g_0^{(i)} = I_n,\ l = 0, 1, \dots, s_i\right\}$. A random selection of the elements of G can be reached by randomly selecting the elements of $U_i, i = 1, 2, \dots, k$ [6-9].

Given a monic polynomial $g(x) \in \mathbb{F}_q[x]$ of degree *n*, the companion matrix is:

$$A = \begin{pmatrix} 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots \\ 0 & 0 & & 1 \\ -g_0 & -g_1 & \cdots & -g_{n-1} \end{pmatrix}$$

where $g_0, g_1, \dots., g_{n-1} \in \mathbb{F}_q$ are the coefficients of $g(x)$. The order $e \in \mathbb{N}$ of matrix A is the smallest positive integer such that $g(x)$ divides $x^e - 1$. From [10] it is known that:

$$(\hat{a}_0, \hat{a}_1, \dots, \hat{a}_{n-1}) = (a_0, a_1, \dots., a_{n-1})A^k$$

is equivalent to

$$(\hat{a}_0 + \hat{a}_1 x + \dots + \hat{a}_{n-1}x^{n-1}) = (a_0 + a_1 x + \dots + a_{n-1}x^{n-1})(x^k)\ mod\ g(x).$$

III. The Schreier structure used in [5]

Let us briefly discuss the Schreier structure analyzed in [5] which consider the base $B = (\beta_1, \beta_2, \dots, \beta_n)$ for $G = GL_n(\mathbb{F}_q)$, where $\beta_i = \left(0, \dots, 0, \underbrace{1}_{i}, 0, \dots, 0\right)$ is the canonical vector with a 1 in position *i*.

Let $L\beta_1 = \begin{bmatrix} \beta_1 & \alpha_{1_1} & \dots & \alpha_{1_{q^n-2}} \\ I_n & A_1 & \dots & A_1 \end{bmatrix}$ and for each $2 \le i \le n$ we define $L\beta_i$ as:

$$\begin{bmatrix} \beta_i & \alpha_{i_1} & \dots & \alpha_{i_{q^{n-i+1}-2}} \\ I_n & A_i & \dots & A_i \\ B_{i_{\alpha^0}} & \alpha_{i_1^0} & \dots & \alpha_{i^0_{q^{n-i+1}-2}} \\ D_{i_{\alpha^0}} & A_i & \cdots & A_i \\ \vdots & \vdots & \dots & \vdots \\ B_{i_{\alpha^{(q-2,\dots,q-2)}}} & \alpha_{i_1^{(q-2,\dots,q-2)}} & \dots & \alpha_{i^{(q-2,\dots,q-2)}_{q^{n-i+1}-2}} \\ D_{i_{\alpha^{(q-2,\dots,q-2)}}} & A_i & \cdots & A_i \end{bmatrix}$$

having the matrix A_i, which stabilizes all the elements of the base before β_i, of the form:

$$A_i = \begin{pmatrix} I_{i-1} & \mathbf{0} \\ \mathbf{0} & A_{i_{n-i+1}} \end{pmatrix} \quad (1)$$

where $A_{i_{n-i+1}}$ is the companion matrix of a primitive polynomial $g_i(x) \in \mathbb{F}_q[x]$ such that $g_i(x) = g_{i,0} + g_{i,1}x + \cdots + g_{i,n-i}x^{n-i+1}$.
In addition, the matrices $D_{i_{\alpha^t}}$ are of the form:

$$D_{i_{\alpha^0}} = \begin{pmatrix} I_{i-1} & \mathbf{0} \\ 0 & \\ \vdots & \\ 0 & A_{i_{n-i+1}} \\ 0 \cdots 0 \quad -g_{i,0}\alpha^0 & \end{pmatrix},$$

$$D_{i_{\alpha^1}} = \begin{pmatrix} I_{i-1} & \mathbf{0} \\ 0 & \\ \vdots & \\ 0 & A_{i_{n-i+1}} \\ 0 \cdots 0 \quad -g_{i,0}\alpha - \alpha^0 & \end{pmatrix},$$

$$D_{i_{\alpha^2}} = \begin{pmatrix} I_{i-1} & \mathbf{0} \\ 0 & \\ \vdots & \\ 0 & A_{i_{n-i+1}} \\ 0 \cdots 0 \quad -g_{i,0}\alpha^2 - \alpha & \end{pmatrix}, \ldots,$$

$$D_{i_{\alpha^{(q^{n-3},\cdots,q^n-2)}}} = \begin{pmatrix} I_{i-1} & \mathbf{0} \\ 0 & \\ \vdots & \\ 0 & A_{i_{n-i+1}} \\ -g_{i,0}(\alpha^{q-3} + \alpha^{q-2}) \quad 0 \cdots 0 & \end{pmatrix}$$

Finally, we have that:

- β_i is the canonical vector with a 1 at the *i-th* coordinate.
- The values of $\beta_{i_{\alpha^t}}$, having a **1** at the *i-th* coordinate are calculated as:

$$\beta_{i_{\alpha^0}} = (0, \ldots, 0, \alpha^0, \underbrace{1}_{i}, 0, \ldots, 0)$$
$$\vdots$$
$$\beta_{i_{\alpha^{q-2}\ldots q-2}} = (\alpha^{q-2}, \ldots, \alpha^{q-2}, \underbrace{1}_{i}, 0, \ldots, 0)$$

- $\alpha_{i_1} = (0, 0, \ldots, 0, \underbrace{1}_{i+1}, 0, \ldots, 0), \ldots, \alpha_{i_1^0} = (0, \ldots, \alpha^0, 0, \underbrace{1}_{i+1}, 0, \ldots, 0), \ldots,$
$\alpha_{i_1^{q-2,\ldots,q-2}} = (\alpha^{q-2}, \ldots, \alpha^{q-2}, 0, \underbrace{1}_{i+1}, 0, \ldots, 0)$
- $\alpha_{i_{q^{n-i+1}-2}} = (0, 0, \ldots, 0, \underbrace{r_{i-1}}_{i}, \ldots, r_{n-2}, -g_{i,n-1}^{-1})$,
$\alpha_{i^0_{q^{n-i+1}-2}} = (0, \ldots, \alpha^0, 0, \underbrace{r_{i-1}}_{i}, \ldots, r_{n-2}, -g_{i,n-1}^{-1}), \ldots,$
$\alpha_{i^{q-3\ldots q-2}_{q^{n-i+1}-2}} = (\alpha^{q-3}, \ldots, \alpha^{q-2}, \underbrace{r_{i-1}}_{i}, \ldots, r_{n-2}, -g_{i,n-1}^{-1})$

having $r_t, i-1 \le t \le n-2$ are arbitrary value of $\mathbb{F}_q$.

IV. Algorithms with the multiplication of two polynomials modulus a monic polynomial with nonzero independent coefficient

From the Schreier structure shown in Section III, the authors of [5] were able to define some algorithms to generate a non-singular matrix A, calculate its inverse A^{-1} and multiply two matrices AB where A and $B \in GL_n(\mathbb{F}_q)$. However, these algorithms use primitive polynomials and when the degree of the same and/or the size of $\mathbb{F}_q$ grows, it is difficult to obtain such polynomials to carry on with the procedure of the algorithms. Such drawback can be avoided by using arbitrary monic polynomials having their independent coefficient distinct from zero.

Theorem 1. Let $B = (\beta_1, \beta_2, \ldots, \beta_n)$ be a basis for $G = GL_n(\mathbb{F}_q)$. the multiplication of a vector $a = (a_0, a_1, \cdots, a_{n-1})$ by $u_i \in U_i$, where U_i is a right transversal or $G^{(i+1)}$ in $G^{(i)}$, $i = 1,2,\cdots,n$, can be expressed as

$$a^{u_1} = (a_0 + a_1x + \cdots + a_{n-1}x^{n-1}) \cdot x^{l_1} \mod v_1(x)$$
$$a^{u_2} = (a_0 + a_1x + \cdots + a_{n-1}x^{n-1}) \cdot x^{l_2} \mod v_2(x)$$
$$= a_0 + C_{2,0} \cdot a_1, (a_1 + a_2x + \cdots + a_{n-1}x^{n-2}) \cdot x^{l_2} \mod v_2(x)$$
$$\vdots$$
$$a^{u_i} = a_0 + C_{i,0} \cdot a_{i-1}, a_1 + C_{i,1} \cdot a_{i-1}, \cdots, a_{i-2} + C_{i,i-2} \cdot a_{i-1}, (a_{i-1} + a_ix + \cdots + a_{n-1}x^{n-i}) \cdot x^{l_i} \mod v_i(x)$$

where $v_i(x) \in \mathbb{F}_q[x]$ is an arbitrary monic polynomial of degree $n-i+1$ having $v_{i,0} \neq 0$ whose order e_i satisfies that $n-i+1 \le e_i \le$

$q^{n-i+1}-1$, $l_i = 0,1,\dots,e_i-1$ with $i = 1,2,\dots,n$ and $C_{i-1,j} \in \mathbb{F}_q$ having $i = 2,\dots,n$ and $j = 0,1,\dots,i-2$.

Proof. To demonstrate the previous theorem let us construct the following Schreier structure. Let $L\beta_1 = \begin{bmatrix} \beta_1 & \alpha_{1_1} & \cdots & \alpha_{1_{e_1-2}} & \delta_{1_1} & \cdots & \delta_{1_{r_1}} \\ I_n & A_1 & \cdots & A_1 & B_1^{\gamma_{1_1}} & \cdots & B_1^{\gamma_{1_{r_1}}} \end{bmatrix}$ be the Schreier structure associated to β_1 where:

- A_1 is the companion matrix of the arbitrary monic polynomial $v_1(x) = v_{1,0} + v_{1,1}x + \cdots + v_{1,n-1}x^{n-1} + x^n \in \mathbb{F}_q[x]$, $v_{1,0} \neq 0$, whose order e_1 satisfies that $n \leq e_1 \leq q^n - 1$.
- $\alpha_{1_1} = (0,1,0,\cdots,0), \cdots, \alpha_{1_{e_1-2}} = (r_0,\cdots,r_{n-2},-g_{1,0}^{-1})$ where $r_t, 0 \leq t \leq n-2$ is an arbitrary value of $\mathbb{F}_q$.
- $\alpha_{1_s}^{A_1^p} \neq \delta_{1_j}, 1 \leq s,p \leq e_1 - 1$ and $1 \leq j \leq r_1$.
- B_1 is the companion matrix of a primitive polynomial $g_1(x) = g_{1,0} + g_{1,1}x + \cdots + g_{1,n-1}x^{n-1} + x^n \in \mathbb{F}_q[x]$.
- $\delta_{1_l} \neq \delta_{1_m}$ for $l \neq m$ having $1 \leq l,m \leq r_1$.
- $\alpha_{1_{e_1-2}}^{B_1} = \delta_{1_1}$ and $\alpha_{1_z}^{B_1} = \delta_{1_{z+1}}$ with $1 \leq z \leq r_1 - 1$.

Later, for $2 \leq i \leq n$ we define $L\beta_i$ as:

$$\begin{bmatrix} \beta_i & \alpha_{i_1} & \dots & \alpha_{i_{e_i-2}} \\ I_n & A_i & \dots & A_i \\ B_{i_{\alpha^0}} & \alpha_{i_1^0} & \dots & \alpha_{i_{e_i-2}^0} \\ D_{i_{\alpha^0}} & A_i & \cdots & A_i \\ \vdots & \vdots & \dots & \vdots \\ B_{i_{\alpha^{(q-2,\dots,q-2)}}} & \alpha_{i_1^{(q-2,\dots,q-2)}} & \dots & \alpha_{i_{e_i-2}^{(q-2,\dots,q-2)}} \\ D_{i_{\alpha^{(q-2,\dots,q-2)}}} & A_i & \cdots & A_i \end{bmatrix}$$

$$\begin{bmatrix} \delta_{i_1} & \delta_{i_2} & \dots & \delta_{i_{r_i}} \\ B_i^{\gamma_{i_1}} & B_i^{\gamma_{i_2}} & \dots & B_i^{\gamma_{i_{r_i}}} \\ \delta_{i_1^0} & \delta_{i_2^0} & \dots & \delta_{i_{q^{n-i+1}-2}^0} \\ B_i^{\gamma_{i_1}} & B_i^{\gamma_{i_2}} & \cdots & B_i^{\gamma_{i_{r_i}}} \\ \vdots & \vdots & & \vdots \\ \delta_{i_1^{(q-2,\dots,q-2)}} & \delta_{i_2^{(q-2,\dots,q-2)}} & \cdots & \delta_{i_{q^{n-i+1}-2}^{(q-2,\dots,q-2)}} \\ B_i^{\gamma_{i_1}} & B_i^{\gamma_{i_2}} & \dots & B_i^{\gamma_{i_{r_i}}} \end{bmatrix}$$

where:

- β_i is the canonical vector with a 1 at the *i-th* coordinate.
- The values of $\beta_{i_{\alpha^t}}$, having a **1** at the *i-th* coordinate are calculated as:

$$\beta_{i_{\alpha^0}} = (0,\dots,0,\alpha^0,\underbrace{1}_{i},0,\dots,0)$$
$$\vdots$$
$$\beta_{i_{\alpha^{q-2\dots q-2}}} = (\alpha^{q-2},\dots,\alpha^{q-2},\underbrace{1}_{i},0,\dots,0)$$

- $\alpha_{i_1} = (0,0,\dots,0,\underbrace{1}_{i+1},0,\dots,0),\dots, \alpha_{i_1^0} = (0,\dots,\alpha^0,0,\underbrace{1}_{i+1},0\dots 0),\dots, \alpha_{i_1^{(q-2,\dots,q-2)}} = \left(\alpha^{q-2},\dots,\alpha^{q-2},0,\underbrace{1}_{i+1},0,\dots,0\right),\cdots, \alpha_{i_{e_i-2}^{(q-2,\dots,q-2)}} = \left(\alpha^{q-2},\dots,\alpha^{q-2},\underbrace{r_{i-1}}_{i-1},\dots,r_{n-2},-g_{i,n-1}^{-1}\right)$
- A_i is a matrix of the form (1) where $A_{i_{n-i+1}}$ is the companion matrix of the arbitrary monic polynomial $v_i(x) \in \mathbb{F}_q[x]$ of degree $n-i+1$ having $v_{i,0} \neq 0$ whose order e_i satisfies that $n-i+1 \leq e_i \leq q^{n-i+1}-1$. The matrix A_i stabilizes all the elements of the base before β_{i-1}.
- $\alpha_{i_s}^{A_i^p} \neq \delta_{i_j}, 1 \leq s,p \leq e_j - 1$ and $1 \leq j \leq r_i$.
- The matrix B_i of the form:

$$B_i = \begin{pmatrix} I_{i-1} & \mathbf{0} \\ \mathbf{0} & B_{i_{n-i+1}} \end{pmatrix}$$

where $B_{i_{n-i+1}}$ has order equal to $q^{n-i+1}-1$ and is the companion matrix of a primitive polynomial $g_i(x) \in \mathbb{F}_q[x]$ of degree $n-i+1$.

- $\delta_{i_l} \neq \delta_{i_m}$ for $l \neq m$ having $1 \leq l,m \leq r_i$.
- $\alpha_{1_{e_1-2}}^{B_i^{\gamma_{i_1}}} = \delta_{i_1}$ and $\alpha_{1_z}^{B_i^{\gamma_{i_{(z+1)}}}} = \delta_{1_{(z+1)}}$ with $1 \leq z \leq r_i - 1$.
- The matrices $D_{i_{\alpha^t}}$ are of the form:

$$D_{i_{\alpha^0}} = B_i^{\gamma_{i_{r_i+1}}} \begin{pmatrix} I_{i-1} & & \mathbf{0} \\ 0 & & \\ \vdots & & \\ 0 & & A_{i_{n-i+1}} \\ 0 \cdots 0 & -v_{i,n-1}\alpha^0 & \end{pmatrix}$$

$$\vdots$$

$$D_{i_{\alpha^{(q-2,\cdots,q-2)}}} = B_i^{\gamma_{i_{r_i+1}}} \begin{pmatrix} I_{i-1} & \mathbf{0} \\ 0 & \\ \vdots & \\ 0 & A_{i_{n-i+1}} \\ -v_{i,n-i}\alpha^{q-3} - \alpha^{q-2} \quad 0 \cdots 0 & \end{pmatrix}$$

- $\delta_{i_{r_i}}^{B_i^{\gamma_{i_{(r+1)_i}}}} = \alpha_{i_{e_i-1}}, \delta_{i_{q^{n-i+1}-2}^0}^{B_i^{\gamma_{i_{(r+1)_i}}}} = \alpha_{i_{e_i-1}^0}, \cdots,$

$\delta_{i_{q^{n-i+1}-2}^{(q-3,\cdots,q-2)}}^{B_i^{\gamma_{i_{(r+1)_i}}}} = \alpha_{i_{e_i-1}^{(q-3,\cdots,q-2)}}$

Then selecting the elements u_i of the right transversal U_i for $G^{(i+1)}$ in $G^{(i)}$, $i = 1,2,\cdots,n$ if such that:

- $u_1 = (A_1)^{l_1}, l_1 = 0,1,\cdots,e_1 - 1$
- For $2 \leq i \leq n, u_i = (A_i)^{l_i}$ or:

$$u_i = (A_i)^{l_i}\left(\prod_{j=1}^{s_0} B_i^{\gamma_{i_j}}\right)\left(D_{i_{\alpha^0}}\right)(A_i)^{l_2\alpha^0}\left(\prod_{j=1}^{s_{i^0}} B_i^{\gamma_{i_j}}\right)\left(D_{i_{\alpha^0}}\right)\cdots\left(D_{i_{\alpha^{x_i}}}\right)(A_i)^{l_{i_{\alpha^{x_i}}}}$$

having $l_2 = l_{i_{\alpha^{x_i}}} = 0,1,\cdots,e_i - 1$ and $x_i = 0,1,\cdots,(q^n - 2,\cdots,q^n - 2)$.

then the multiplication of a vector $a = (a_0, a_1, \cdots, a_{n-1})$ by u_i can be expressed as:

$$a^{u_1} = (a_0 + a_1x + \cdots + a_{n-1}x^{n-1}) \cdot x^{l_1} \mod v_1(x)$$

$$a^{u_2} = (a_0 + a_1x + \cdots + a_{n-1}x^{n-1}) \cdot x^{l_2} \mod v_2(x)$$

$$= a_0 + C_{2,0} \cdot a_1, (a_1 + a_2x + \cdots + a_{n-1}x^{n-2}) \cdot x^{l_2} \mod v_2(x)$$

$$\vdots$$

$$a^{u_i} = a_0 + C_{i,0} \cdot a_{i-1}, a_1 + C_{i,1} \cdot a_{i-1}, \cdots, a_{i-2} + C_{i,i-2} \cdot a_{i-1}, (a_{i-1} + a_ix + \cdots + a_{n-1}x^{n-i}) \cdot x^{l_i} \mod v_i(x)$$

Therefore, the proof is complete.

An example of the development of the Schreier structure used to demonstrate Theorem 1 is shown in Appendix A.

Using the above reasoning, we obtain a method for generating a non-singular matrix based in the multiplication of polynomials modulus a monic polynomial with non-zero independent coefficient as shown in the next algorithm.

Algorithm 1: Generate a matrix $A \in Subset\left(GL_n(\mathbb{F}_q)\right)$

Require: $C_{i-1,j} \in \mathbb{F}_q$, $i = 2, \ldots, n$ and $j = 0,1, \ldots, i-2$.
Require: Monic polynomials $v_i(x) = v_{i,0} + v_{i,1}x + \cdots + v_{i,n-1}x^{n-i} + x^{n-i+1} \in \mathbb{F}_q[x]$, $v_{i,0} \neq 0, 1 \leq i \leq n$
Require: $l_i = 0,1, \ldots, e_i - 1$ where e_i is the order of $v_i(x)$
Ensure: A nonsingular matrix $A \in Subset\left(GL_n(\mathbb{F}_q)\right)$

// Calculate the first row of A (r_1)
$(a_0, a_1, \ldots., a_{n-1}) = (1, 0, \ldots, 0)$
$a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$
$(\hat{a}_0 + \hat{a}_1x + \cdots + \hat{a}_{n-1}x^{n-1}) = \big(a(x)\big)(x^{l_1}) \mod v_1(x)$
$\boldsymbol{r_1} = (\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{n-1})$

// Calculate the *j-th* row of A ($2 \leq j \leq n$)
for $j = 2$ **to** n **do:**
  $(a_0, a_1, \ldots., a_{n-1}) = \big(0, 0, \ldots, a_{j-1} = 1, 0, \ldots, 0\big)$
  for $i = j$ **down to** 2 **do:**
    $\hat{a}_0 = a_0 + C_{i,0} \cdot a_{i-1}$
    $\hat{a}_1 = a_1 + C_{i,1} \cdot a_{i-1}$
    $\hat{a}_2 = a_2 + C_{i,2} \cdot a_{i-1}$
    $\vdots$
    $\hat{a}_{i-2} = a_{i-2} + C_{i,i-2} \cdot a_{i-1}$
    $a(x) = a_{i-1} + a_ix + \cdots + a_{n-1}x^{n-i}$
    $(\hat{a}_{i-1} + \cdots + \hat{a}_{n-1}x^{n-1}) = \big(a(x)\big)(x^{l_i}) \mod v_i(x)$
    $(a_0, a_1, \ldots., a_{n-1}) = (\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{n-1})$
  end for
  $a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$
  $(\hat{a}_0 + \cdots + \hat{a}_{n-1}x^{n-1}) = \big(a(x)\big)(x^{l_1}) \mod v_1(x)$
  $\boldsymbol{r_j} = (\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{n-1})$
end for

It worth to remark that using monic polynomials we cannot generate the whole $GL_n(\mathbb{F}_q)$. For the subset generated by using these polynomials we will use the notation $Subset\left(GL_n(\mathbb{F}_q)\right)$ and its cardinality is given in the following proposition.

Proposition 1. The cardinality of $Subset\left(GL_n(\mathbb{F}_q)\right)$, denoted by $\#Subset\left(GL_n(\mathbb{F}_q)\right)$ is equal to

$$\#Subset\left(GL_n(\mathbb{F}_q)\right) = \left(\prod_{i=1}^{n-1} q^i\right)\left(\prod_{j=1}^{n} q^{e_j} - 1\right)$$

Proof. The cardinality of $Subset\left(GL_n(\mathbb{F}_q)\right)$ is the consequence of the construction of the Schreier structure used to demonstrate Theorem 1.

Analogously to the method presented in Algorithm 1 one can deduce the procedures to obtain the inverse of a matrix $A \in$

$Subset\left(GL_n(\mathbb{F}_q)\right)$ and to multiply a vector for a matrix A or its inverse A^{-1}. The pseudo-codes of these algorithms are presented in Appendix B.

V. Algorithms for matrices with variable right transversal U_i

Let us consider that the values l_i will always be the same regardless the remaining input parameters of Algorithm 1. If one sets up $l_i = n - i + 1$ having $i = 1,2,\cdots,n$ then the algorithm is still able to generate a matrix $A \in Subset\left(GL_n(\mathbb{F}_q)\right)$. In addition, let us enunciate the following proposition.

Proposition 2. Let $v_i(x) = v_{i,0} + v_{i,1}x + \cdots + v_{i,n-1}x^{n-i} + x^{n-i+1} \in \mathbb{F}_q[x]$ and $h_i(x) = h_{i,0} + h_{i,1}x + \cdots + h_{i,n-1}x^{n-i} + x^{n-i+1} \in \mathbb{F}_q[x]$ having $h_{i,0}, v_{i,0} \neq 0$ *for* $1 \leq i \leq n$ two inputs of Algorithm 1 and let $l_i = n - i + 1$ for $i = 1,2,\cdots,n$. If for some value of i it is true that $v_i(x) \neq h_i(x)$ then the algorithm will produce two distinct matrices.

Proof. Let $v_1(x) \neq h_1(x)$ and let us assume that the first row of each matrix are equal. Since the first row of a matrix generated using Algorithm 1 is obtained as the result of $\beta_1^{u_1}$ then two arbitrary matrices produced by the method will have an equal first row if and only if $v_1(x) = h_1(x)$. Hence, when $v_1(x) \neq h_1(x)$ the first row of both matrices are different.

For $2 \leq i \leq n$, let $v_i(x) \neq h_i(x)$ and the first $i - 1$ rows of both matrices be equal and let us assume that the *i-th* row of the matrices are also equal. Since the *i-th* row is obtained as $r_i = \beta_1^{u_i u_{i-1} \cdots u_1}$ then it must be true that the values $C_{k,j}$ on both matrices are equal as well as the polynomials $v_i(x)$ and $h_i(x)$. However, $v_i(x) \neq h_i(x)$, therefore the *i-th* row of both matrices are different.

As result of Proposition 2 we obtain the number of distinct matrices of $GL_n(\mathbb{F}_q)$ that can be generated using Algorithm 1 when $l_i = n - i + 1$ for $i = 1,2,\cdots,n$ as stated in the following corollary.

Corollary 1. The number of distinct matrices of $GL_n(\mathbb{F}_q)$ that can be generated using Algorithm 1 when $l_i = n - i + 1$ for $i = 1,2,\cdots,n$ is:

$$\left(\prod_{i=1}^{n-1} q^i\right)^2 (q-1)^n$$

Finally, if we take under consideration the following set of input parameters:

- $C_{i-1,j} \in \mathbb{F}_q, i = 2,3,\cdots,n$ and $j = 0,1,\cdots,i-2$.

- Monic polynomials $v_i(x) = v_{i,0} + v_{i,1}x + \cdots + v_{i,n-1}x^{n-i} + x^{n-i+1} \in \mathbb{F}_q[x]$, $v_{i,0} \neq 0$ where $v_i(x)$ is even and $2 \leq i \leq n$. If $\mathbb{F}_q = \mathbb{F}_{2^s}$ then $v_{1,0} = 1$ else $v_{1,0} = 1$ or $v_{n,0} = \frac{q-1}{2}$.

- $l_i = x^{\frac{e_i}{2}}$ where e_i is the order of $v_i(x)$ and $i = 1,2,\cdots,n$.

then Algorithm 1 is able to generate an involutory matrix $A \in Subset\left(GL_n(\mathbb{F}_q)\right)$.

VI. Computational complexity of the algorithms

In the preceding section we discuss the definition of new algorithms for generating non-singular matrices as result of the multiplication of polynomials modulus a monic polynomial with non-zero independent coefficient. The new algorithms avoid the use of primitive polynomials which can be a drawback as the size of $\mathbb{F}_q$ or parameter *n* increases. As result of this trade-off, the number of matrices that can be generated by these algorithms was reduced as stated in Proposition 1. However, the use of monic polynomials instead of primitive ones does not have any implication towards the time complexity of the algorithms since we only substitute the primitive polynomials from the original version of the same presented in [5] by monic polynomials of identical degree. Thus, any complexity analysis we conduct over the algorithms described in this paper can be extended to those presented in [5].

It is well-known that the naive polynomial multiplication algorithm is not the optimal way to multiply two polynomials. There are several research papers where this topic is studied and to the best of our knowledge the complexity of multiplying two polynomials of degree less than n in $\mathbb{F}_q[x]$ is given in [11]. However, to show that the bound given in [5] is not tight we will use the same complexity formula related to the multiplication of polynomials the authors have used, which is equals to $\mathcal{O}(n \log n \log\log n)$ according to [12].

In [5], the authors assume that every polynomial multiplication carried by their methods have complexity $\mathcal{O}(n \log n \log\log n)$ which is not true since they work with polynomials of degree strictly less than n to calculate the intermediate values necessary to construct each row of the matrix. In addition, let us consider the following:

- The polynomials used within the algorithm are of degree at most $n - i + 1$, $i = 1,2,\cdots,n$ with coefficients in $\mathbb{F}_q[x]$.

- The complexity of multiplying two elements on $\mathbb{F}_q$, denoted $\mathcal{M}_q$, is the same regardless the degree of the polynomial $v(x)$ from which they are coefficients and it is $\mathcal{O}(b \log b \log\log b)$ where b is the size in bits of the same.

- We denote by $\mathcal{C}_i = \mathcal{O}(i \log i \log\log i)$ the complexity of multiplying two polynomials module an i-degree polynomial.

Proposition 3. The computational complexity of Algorithm 1 is

$$T(n) = \sum_{i=1}^{n}(i-1)\mathcal{C}_i + \mathcal{M}_q \sum_{i=1}^{n-2}\frac{i(i+1)}{2}$$

Proof. For demonstrating the proposition, let us show the number of polynomial and finite field multiplications carried by the algorithm on each step according to the Schreier structure defined in Section V.

Calculation of the i-th row	Polynomial multiplications	Finite field multiplications
β_1	-	-
$(\beta_2)^{u_1}$	$\mathcal{C}_n$	-
$(\beta_3)^{u_2u_1}$	$\mathcal{C}_n + \mathcal{C}_{n-1}$	1
⋮	⋮	⋮
$(\beta_i)^{u_{i-2}u_{i-1}\cdots u_1}$	$\mathcal{C}_n + \cdots + \mathcal{C}_{n-i+2}$	$i-2+\cdots+1$
⋮	⋮	⋮
$(\beta_{n-1})^{u_{n-2}u_{n-3}\cdots u_1}$	$\mathcal{C}_n + \cdots + \mathcal{C}_3$	$n-3+\cdots+1$
$(\beta_n)^{u_{n-1}u_{n-2}\cdots u_1}$	$\mathcal{C}_n + \cdots + \mathcal{C}_2$	$n-2+\cdots+1$

From the above decomposition we obtain that the complexity of all the multiplications of two polynomials modulus an i–degree polynomial defined over $\mathbb{F}_q$ where $i = 2,3,\cdots,n$ is given by the formula

$$((n-1)\mathcal{C}_n + (n-2)\mathcal{C}_{n-1} + \cdots + 2\mathcal{C}_3 + \mathcal{C}_2)$$

which can be simply written as:

$$\sum_{i=1}^{n}(i-1)\mathcal{C}_i$$

In addition, the amount of element multiplications in $\mathbb{F}_q$ whose computational complexity is $\mathcal{M}_q$ is equal to:

$$\sum_{i=1}^{n-2} i + \sum_{i=1}^{n-3} i + \cdots + \sum_{i=1}^{2} i + 1 = \sum_{i=1}^{n-2}\frac{i(i+1)}{2}$$

Then, it is straightforward to see that the computational complexity of the algorithm is:

$$\sum_{i=1}^{n}(i-1)\mathcal{C}_i + \mathcal{M}_q \sum_{i=1}^{n-2}\frac{i(i+1)}{2}$$

Therefore, the proof is complete.

In [5] the total computational complexity of multiplying two elements in $\mathbb{F}_q$ was deemed to be constant and therefore removed from the complexity expression of the algorithm. However, as one can see in the proof of Proposition 3, the number of element multiplications in $\mathbb{F}_q$ increases by quadratic order as the value of n does. Thus, one cannot simply discard such value when calculating the complexity of the algorithm. Moreover, as we aforementioned the authors of [5] were absolute to say that all multiplication of polynomials modulus an i-degree polynomial had

computational complexity of $\mathcal{C}_n$ which we have shown is not the case when the module polynomial have degree strictly lower than *n*. This leads us to conclude that a more accurate upper bound for the algorithm presented in [5] for generating a non-singular matrix is given by the result of Proposition 3. The remaining of this section is dedicated to address the complexity of the algorithms presented in Appendix A.

It is known that the complexity of finding the inverse of a polynomial modulus an i–degree polynomial over $\mathbb{F}_q$ has complexity $\mathcal{O}(\mathcal{C}_i \log i)$ [13]. This yields in the following proposition.

Proposition 4. The computational complexity of Algorithm 2 is

$$T(n) = \sum_{i=1}^{n}(i-1)\mathcal{C}_i + \mathcal{M}_q \sum_{i=2}^{n-1}\frac{i(i+1)}{2} + \sum_{i=1}^{n}\mathcal{C}_i \log i + (n-1)\mathcal{C}_n$$

Proof. To prove the above proposition, we follow the same reasoning as for Proposition 3. Let us show the number of polynomial and element multiplications carried by the algorithm.

Calculation of the i-th row	Polynomial multiplications	Finite field multiplications
$\beta_1^{u_1^{-1}}$	-	-
$(\beta_2)^{u_1^{-1}u_2^{-1}}$	$\mathcal{C}_n + \mathcal{C}_{n-1}$	1
$(\beta_3)^{u_1^{-1}u_2^{-1}u_3^{-1}}$	$\mathcal{C}_n + \mathcal{C}_{n-1} + \mathcal{C}_{n-2}$	2+1
⋮	⋮	⋮
$(\beta_i)^{u_1^{-1}\cdots u_{i-1}^{-1}u_i^{-1}}$	$\mathcal{C}_n + \cdots + \mathcal{C}_{n-i+1}$	$i-1+\cdots+1$
⋮	⋮	⋮
$(\beta_{n-1})^{u_1^{-1}\cdots u_{n-2}^{-1}u_{n-1}^{-1}}$	$\mathcal{C}_n + \cdots + \mathcal{C}_2$	$n-2+\cdots+1$
$(\beta_n)^{u_1^{-1}\cdots u_{n-1}^{-1}u_n^{-1}}$	$\mathcal{C}_n + \cdots + \mathcal{C}_1$	$n-1+\cdots+1$

From the above decomposition we obtain that the complexity of all the multiplications of two polynomials modulus an i–degree polynomial defined over $\mathbb{F}_q$ where $i = 2,3,\cdots,n$ is given by the formula

$$((n-1)\mathcal{C}_n + (n-1)\mathcal{C}_{n-1} + \cdots + 2\mathcal{C}_2 + \mathcal{C}_1)$$

which can be simply written as:

$$\sum_{i=1}^{n-1} i \cdot \mathcal{C}_i + (n-1)\mathcal{C}_n$$

In addition, the amount of element multiplications in $\mathbb{F}_q$ whose computational complexity is $\mathcal{M}_q$ is equal to:

$$\sum_{i=1}^{n-1} i + \sum_{i=1}^{n-2} i + \cdots + \sum_{i=1}^{2} i + 1 = \sum_{i=1}^{n-1}\frac{i(i+1)}{2}$$

Later, for calculating the i-th row of the matrix the algorithm performs a polynomial inversion modulus an $(n-i+1)$-degree polynomial, and the complexity of making all polynomial inversions is given by the formula:

$$\sum_{i=1}^{n}\mathcal{C}_i \log i$$

Then, it is straightforward to see that the computational complexity of Algorithm 2 is equal to:

$$\sum_{i=1}^{n}(i-1)\mathcal{C}_i + \mathcal{M}_q \sum_{i=2}^{n-1}\frac{i(i+1)}{2} + \sum_{i=1}^{n}\mathcal{C}_i \log i + (n-1)\mathcal{C}_n$$

Therefore, the proof is complete.

Although the result of Proposition 4 encompasses all the operations made within the algorithm, by applying the rule of the sum for algorithmic complexity we can discard the two rightmost members of the formula given in the proposition. Hence we can summarize the complexity of Algorithm 2 in the following corollary.

Corollary 2. The computational complexity of Algorithm 2 is

$$\mathcal{O}\left(\sum_{i=1}^{n}(i-1)\mathcal{C}_i + \mathcal{M}_q \sum_{i=2}^{n-1}\frac{i(i+1)}{2}\right)$$

Finally, let us discuss the complexity of multiplying a row vector by a matrix or its inverse. The next two propositions give the complexity bound for the same.

Proposition 5. The computational complexity of Algorithm 3 is

$$T(n) = \sum_{i=1}^{n} \mathcal{C}_i \; + \mathcal{M}_q \frac{n(n-1)}{2}$$

Proof. Let us show the number of polynomial and field element multiplications in Algorithm 3.

Calculation of the result vector
$(a_0, a_1, \cdots, a_{n-1})^{u_n u_{n-1} \cdots u_1}$
Polynomial multiplications
$\mathcal{C}_n + \mathcal{C}_{n-1} + \cdots + \mathcal{C}_2 + \mathcal{C}_1$
Finite field multiplications
$n-1+n-2+\cdots+2+1$

From the above decomposition is easy to check that the total complexity of multiplying two polynomials modulus an i-degree polynomial defined over $\mathbb{F}_q$ is:

$$(\mathcal{C}_n + \mathcal{C}_{n-1} + \cdots + \mathcal{C}_2 + \mathcal{C}_1) = \sum_{i=1}^{n} C_i$$

whereas the complexity of multiplying elements of $\mathbb{F}_q$ is:

$$\mathcal{M}_q \cdot \sum_{i=1}^{n-1} i = \mathcal{M}_q \frac{n(n-1)}{2}$$

Hence, the complexity of the algorithm is

$$\sum_{i=1}^{n} \mathcal{C}_i \; + \mathcal{M}_q \frac{n(n-1)}{2}$$

Therefore, the proof is complete.

□

Proposition 6. The computational complexity of Algorithm 4 is

$$T(n) = \sum_{i=1}^{n} \mathcal{C}_i \log i \; + \sum_{i=1}^{n} \mathcal{C}_i \; + \mathcal{M}_q \frac{n(n-1)}{2}$$

Proof. The demonstration of the proposition is identical to the one for Proposition 5 but we also have to take into account the number of polynomial inversions made by the algorithm whose complexity is equal to:

$$(\mathcal{C}_n \log n + \mathcal{C}_{n-1} \log(n-1) + \cdots + \mathcal{C}_1 \log 1) = \sum_{i=1}^{n} C_i \log i$$

which yields in that the complexity of the algorithm is

$$\sum_{i=1}^{n} \mathcal{C}_i \log i \; + \sum_{i=1}^{n} \mathcal{C}_i \; + \mathcal{M}_q \frac{n(n-1)}{2}$$

This completes the proof.

To this point we have shown several algorithms related to the generation of non-singular matrices with coefficients over $\mathbb{F}_q$ and the multiplication of a row vector by these matrices as well as discussed the computational complexity of the same. We like to remark that although we do not improve the time complexity of the algorithm introduced in [4] our algorithms use, at most, n random elements of $\mathbb{F}_q$ while the one presented in [4] uses $n^2 + \mathcal{O}(1)$ random field elements in average. Furthermore, in order to multiply a row vector by a matrix or its inverse we do not have to represent the matrix to make the calculations.

VII. Reducing the Complexity of Generating a Non-Singular Matrix and Its Inverse

For the case of general matrices there exist several research papers that propose methods to obtain their inverses [14–16]. Other algorithms focus on special types of matrices such as positive definite matrix [17], tridiagonal matrix [18] and diagonal matrix [19]. However, such algorithms have computational complexity of $\mathcal{O}(n^3)$. The first method to have a complexity strictly lower than $\mathcal{O}(n^3)$ was proposed by Strassen in [20] having a running time of $\mathcal{O}(n^{2.808})$ and, after several improvement to the index of n, Coppersmith and Winograd obtain a method capable of finding the inverse of a matrix in $\mathcal{O}(n^{2.496})$ [21]. There exists methods whose running time is $\mathcal{O}(n^2)$ such as the proposed by Traub for inversion of classical Vandermonde matrices [22], which has since been extended to many important cases beyond the classical Vandermonde case. Nonetheless, we do not find any reference where the

generation of the inverse matrix take less than $\mathcal{O}(n^2)$ operations. Through this section we discuss some input configurations for the algorithms discussed in this paper which allow to reduce their computational complexity.

Proposition 7. *Let* $C_{i-1,j} = 0,\ i = 2,3,\cdots,n,\ j = 0,1,\cdots,i-2$ *and let* $l_1 \neq 0$ *while* $l_i = 0\ \forall i > 1$. *Then, the complexity of Algorithm 1 is*

$$T(n) = (n-1)\mathcal{C}_n$$

Proof. Notice that if $C_{i,j} = 0$ then no field element multiplications are carried by Algorithm 1. Furthermore, if $\forall i > 1$ we have $l_i = 0$ then the operation

$$a_i(x) \cdot x^{l_i} \ \ mod\ v_i(x)$$

where $a_i(x) = a_{i-1} + a_i x + \cdots + a_{n-1}x^{n-i}$ can be reduced to

$$a_i(x) \ \ mod\ v_i(x)$$

which is equal to $a_i(x)$ given that such polynomial have degree lower than $v_i(x)$. Thus, no complexity is added for the multiplication of polynomials modulus $v_i(x)$ when $i > 1$. Later, to construct the *j-th* row is only necessary to operate:

$$(a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}) \cdot x^{l_1} \ \ mod\ \ v_1(x)$$

whose complexity is $\mathcal{C}_n$. Hence, it is easy to check that complexity of Algorithm 1 in this case is equal to $(n-1)\mathcal{C}_n$.

From the result of the above proposition one can obtain the following corollary.

Corollary 3. Let $C_{i-1,j} = 0,\ i = 2,3,\cdots,n,\ j = 0,1,\cdots,i-2$ and *let* $l_1 \neq 0$ while $l_i = 0\ \forall i > 1$. Then, the complexity of Algorithm 1 is $\mathcal{O}(n^2 \log n \log\log n)$.

Remark 1. It worth noticing that for matrix sizes of order of thousands the number of operations made by the new configuration of Algorithm 1 is lower than the one presented by Coppersmith and Winograd [21].

Proposition 8. Let $C_{i-1,j} = 0,\ i = 2,3,\cdots,n,\ \ j = 0,1,\cdots,i-2$ and *let* $l_1, l_2 \neq 0$ *while* $l_i = 0\ \forall i > 2$. Then, the complexity of Algorithm 1 is $\mathcal{O}(n^2 \log n \log\log n)$.

Proof. Using an analogous reasoning to proof of Proposition 7 one can easily obtain that the complexity of the algorithm is given by the formula

$$T(n) = (n-1)\mathcal{C}_n + (n-2)\mathcal{C}_{n-1}$$

from where it is straightforward to notice that

$$T(n) < 2n\mathcal{C}_n \in \mathcal{O}(n^2 \log n \log\log n)$$

This completes the proof.

The results from Propositions 7 and 8 can be generalized in the following theorem.

Theorem 2. Let $C_{i-1,j} = 0,\ i = 2,3,\cdots,n,$ $j = 0,1,\cdots,i-2$ and let $l_1, l_2, \cdots, l_i \neq 0$ while $l_k = 0\ \forall k > i$. Then, the complexity of Algorithm 1 is $\mathcal{O}(i \cdot n^2 \log n \log\log n)$.

Proof. It is the consequence of the combination of the proofs for Propositions 3, 7 and 8.

Notice that, as result of Theorem 2, when *i = n* then the algorithm running time is roughly bounded by $\mathcal{O}(n^3 \log n \log\log n)$ which correspond to the bound given in [5]. Furthermore, the more $l_i \neq 0$ we use the greater is the subspace of matrices we can generate by means of our proposal.

In the next table we show a comparison of the result from Theorem 2 and other methods to obtain non-singular matrices in the literature.

Reference	**Type of matrix**	**Upper bound for complexity**
Theorem 2	Arbitrary	$\mathcal{O}(i \cdot n^2 \log n \log\log n)$
Ref. [17]	Positive definite	$\mathcal{O}(n^3)$
Ref. [18]	Tridiagonal	$\mathcal{O}(n^3)$
Ref. [19]	Diagonal	$\mathcal{O}(n^3)$
Ref. [20]	Arbitrary	$\mathcal{O}(n^{2,808})$
Ref. [21]	Arbitrary	$\mathcal{O}(n^{2,496})$
Ref. [22]	Classical Vandermonde	$\mathcal{O}(n^2)$

Once we have analyzed special input cases of the algorithm to obtain nonsingular matrices in less than $\mathcal{O}(n^3)$ operations let us show that the complexity of multiplying a row vector by such matrices is equivalent to multiplication of polynomials modulus an *n*-degree polynomial over $\mathbb{F}_q$.

Proposition 9. Let $C_{i-1,j} = 0,\ i = 2,3,\cdots,n,\ j = 0,1,\cdots,i-2$ and let $l_1 \neq 0$ while $l_i = 0\ \forall i > 1$. Then, the complexity of Algorithm 3 is

$$T(n) = \mathcal{C}_n \in \mathcal{O}(n \log n \log\log n)$$

Proof. It worth noticing that in Algorithm 3 the input vector $\vec{a} = (a_0, a_1, \cdots, a_{n-1})$ remains constant regardless the value of $i = n, n-1, \cdots, 3,2$ given by the fact that both $C_{i,j}$ and l_i are equal to 0. Hence the result of the multiplication of $\vec{a}$ by the non-singular matrix A is equivalent to

$$(a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}) \cdot x^{l_1} \mod v_1(x)$$

whose complexity is $\mathcal{C}_n \in \mathcal{O}(n \log n \log\log n)$. □

Following the reasoning of Propositions 7 and 8 which yield in Theorem 2 we can also generalize the result of the above proposition as stated in the following corollary.

Corollary 4. Let $C_{i-1,j} = 0,\ i = 2,3,\cdots,n,\ j = 0,1,\cdots,i-2$ and let $l_1, l_2, \cdots, l_i \neq 0$ while $l_k = 0\ \forall k > i$. Then, the complexity of Algorithm 3 is $\mathcal{O}(i \cdot n \log n \log\log n)$.

In addition, we believe that the prior knowledge of the factorization of the monic polynomials will also have influence on the complexity of the algorithms, but such is a topic we will board in a future research.

VIII. Conclusions

In this paper several methods for the construction of non-singular matrices were studied. We show that one can substitute the primitive polynomials used in [5] by monic polynomials with non-zero independent coefficient and yet generate non-singular matrices without affect the complexity of the algorithm. In this fashion we provide the proofs of the complexity bounds of the methods presented here which are more accurate than the ones in [5]. In addition, we analyze some special input configurations which allow to reduce the complexity of the algorithm to generate non-singular matrices to $\mathcal{O}(n^2 \log n \log\log n)$ and the algorithm to multiply a row vector by a non-singular matrix to $\mathcal{O}(n \log n \log\log n)$. Finally, we generalize the results for a given configuration of the input of the algorithms which take $C_{i-1,j} = 0,\ i = 2,3,\cdots,n,\quad j = 0,1,\cdots,i-2,\ l_1, l_2, \cdots, l_s \neq 0$ *while* $l_k = 0\ \forall k > s$ for some $s = 1,2,\cdots,n$.

Appendix A. Example of the proposed Schreier structure

Let

$$A_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

be the companion matrix of the arbitrary monic polynomial

$$v_1(x) = x^4 + x^3 + x^2 + x + 1 \ \in \mathbb{F}_2[x]$$

with order 5 and let

$$B_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

be the companion matrix of the primitive polynomial $b_1(x) = x^4 + x + 1$. Then L_{β_1} will be:

$$(1,\ 0,\ 0,\ 0)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (0,\ 1,\ 0,\ 0)\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad (0,\ 0,\ 1,\ 0)\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$(0,\ 0,\ 0,\ 1)\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad (1,\ 1,\ 1,\ 1)\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad (1,\ 0,\ 1,\ 1)\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$(1,\ 0,\ 0,\ 1)\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad (1,\ 1,\ 0,\ 0)\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad (0,\ 1,\ 1,\ 0)\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}^5$$

$$\begin{matrix}(0,&0,&1,&1)\\ \begin{pmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&1&0&0\end{pmatrix}\end{matrix}\quad\begin{matrix}(1,&1,&0,&1)\\ \begin{pmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&1&0&0\end{pmatrix}\end{matrix}\quad\begin{matrix}(1,&0,&1,&0)\\ \begin{pmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&1&0&0\end{pmatrix}\end{matrix}$$

$$\begin{matrix}(0,&1,&0,&1)\\ \begin{pmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&1&0&0\end{pmatrix}\end{matrix}\quad\begin{matrix}(1,&1,&1,&0)\\ \begin{pmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&1&0&0\end{pmatrix}\end{matrix}\quad\begin{matrix}(0,&1,&1,&1)\\ \begin{pmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&1&0&0\end{pmatrix}\end{matrix}$$

Let

$$A_2=\begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&1&1\end{pmatrix},\qquad A_2'=\begin{pmatrix}0&1&0\\0&0&1\\1&1&1\end{pmatrix}$$

where A_2' is the companion matrix of the polynomial $v_2(x)=1+x+x^2+x^3$ with order 4 and let

$$B_2=\begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&1\end{pmatrix},\qquad B_2'=\begin{pmatrix}0&1&0\\0&0&1\\1&0&1\end{pmatrix}$$

where B_2' is the companion matrix of the primitive polynomial $b_2(x)=1+x^2+x^3$. Then L_{β_2} will be:

$$\begin{matrix}(0,&1,&0,&0)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(0,&0,&1,&0)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&1&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(0,&0,&0,&1)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&1&1\end{pmatrix}\end{matrix}$$

$$\begin{matrix}(0,&1,&1,&1)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&1&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(0,&1,&1,&0)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(0,&0,&1,&1)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&1\end{pmatrix}\end{matrix}$$

$$\begin{matrix}(0,&1,&0,&1)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&1\end{pmatrix}\end{matrix}^4\quad\underbrace{\begin{matrix}((0,&1,&1,&1)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(1,&1,&0,&0))\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\1&1&1&1\end{pmatrix}\end{matrix}}_{D_{2_1}}$$

$$\begin{matrix}(1,&1,&0,&0)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&1&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(1,&0,&1,&0)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&1&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(1,&0,&0,&1)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&1&1\end{pmatrix}\end{matrix}$$

$$\begin{matrix}(1,&1,&1,&1)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&1&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(1,&1,&1,&0)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(1,&0,&1,&1)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&1\end{pmatrix}\end{matrix}$$

$$\begin{matrix}(1,&1,&0,&1)\\ \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&1\end{pmatrix}\end{matrix}$$

Let

$$A_3=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix},\qquad A_3'=\begin{pmatrix}0&1\\1&1\end{pmatrix}$$

where A_3' is the companion matrix of the polynomial $v_3(x)=1+x+x^2$ with order 3. Since the polynomial $v_3(x)$ is primitive, the matrix B_3 is not necessary. Then L_{β_3} will be:

$$\begin{matrix}(0,&0,&1,&0)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(0,&0,&0,&1)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(0,&0,&1,&1)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix}\end{matrix}$$

$$\underbrace{\begin{matrix}(0,&1,&1,&0)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&1&1&1\end{pmatrix}\end{matrix}}_{D_{3_1}}\quad\begin{matrix}(0,&1,&0,&1)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(0,&1,&1,&1)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix}\end{matrix}$$

$$\underbrace{\begin{matrix}(1,&0,&1,&0)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&1&1&1\end{pmatrix}\end{matrix}}_{D_{3_2}}\quad\begin{matrix}(1,&0,&0,&1)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(1,&0,&1,&1)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix}\end{matrix}$$

$$\underbrace{\begin{matrix}(1,&1,&1,&0)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&1&1&1\end{pmatrix}\end{matrix}}_{D_{3_3}}\quad\begin{matrix}(1,&1,&0,&1)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix}\end{matrix}\quad\begin{matrix}(1,&1,&1,&1)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix}\end{matrix}$$

Finally, taking $v_4(x)=1+x$ then L_{β_4} will be:

$$\begin{matrix}(0,&0,&0,&1)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\end{matrix}\quad\underbrace{\begin{matrix}(0,&0,&1,&1)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix}\end{matrix}}_{D_{4_1}}\quad\underbrace{\begin{matrix}(0,&0,&1,&1)\\ \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&1&1&1\end{pmatrix}\end{matrix}}_{D_{4_2}}$$

$$\underbrace{\begin{matrix}(0, & 1, & 1, & 1)\\ \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 1\end{pmatrix}\end{matrix}}_{D_{43}} \quad \underbrace{\begin{matrix}(1, & 0, & 0, & 1)\\ \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 1 & 1 & 1\end{pmatrix}\end{matrix}}_{D_{44}} \quad \underbrace{\begin{matrix}(1, & 0, & 1, & 1)\\ \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 1\end{pmatrix}\end{matrix}}_{D_{45}}$$

$$\underbrace{\begin{matrix}(1, & 1, & 0, & 1)\\ \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 1 & 1 & 1\end{pmatrix}\end{matrix}}_{D_{46}} \quad \underbrace{\begin{matrix}(1, & 1, & 1, & 1)\\ \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 1 & 1 & 1\end{pmatrix}\end{matrix}}_{D_{47}}$$

APPENDIX B. ALGORITHMS

Algorithm 2: Generate the matrix A^{-1}

Require: $C_{i-1,j} \in \mathbb{F}_q$, $i = 2, \dots, n$ and $j = 0,1, \dots, i-2$.
Require: Monic polynomials $v_i(x) = v_{i,0} + v_{i,1}x + \cdots + v_{i,n-1}x^{n-i} + x^{n-i+1} \in \mathbb{F}_q[x]$, $v_{i,0} \neq 0, 1 \leq i \leq n$
Require: $l_i = 0,1, \dots, e_i - 1$ where e_i is the order of $v_i(x)$
Ensure: The matrix A^{-1}

// Calculate the *j-th* row of A^{-1} ($1 \leq j \leq n$)
for $j = 1$ **to** n **do:**
  $(a_0, a_1, \dots., a_{n-1}) = (0, 0, \dots, a_{j-1} = 1, 0, \dots, 0)$
  $a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$
  $(\hat{a}_0 + \cdots + \hat{a}_{n-1}x^{n-1}) = \big(a(x)\big)(x^{-l_1}) \; mod \; v_1(x)$
  $(a_0, a_1, \dots., a_{n-1}) = (\hat{a}_0, \hat{a}_1, \dots, \hat{a}_{n-1})$
  for $i = j$ **down to** 2 **do:**
    $\hat{a}_0 = a_0 - C_{i,0} \cdot a_{i-1}$
    $\hat{a}_1 = a_1 - C_{i,1} \cdot a_{i-1}$
    $\hat{a}_2 = a_2 - C_{i,2} \cdot a_{i-1}$
    ⋮
    $\hat{a}_{i-2} = a_{i-2} + C_{i,i-2} \cdot a_{i-1}$
    $a(x) = a_{i-1} + a_ix + \cdots + a_{n-1}x^{n-i}$
    $(\hat{a}_{i-1} + \cdots + \hat{a}_{n-1}x^{n-1}) = \big(a(x)\big)(x^{-l_i}) \; mod \; v_i(x)$
    $(a_0, a_1, \dots., a_{n-1}) = (\hat{a}_0, \hat{a}_1, \dots, \hat{a}_{n-1})$
  end for
  $\boldsymbol{r_j} = (a_0, a_1, \dots., a_{n-1})$
end for

Algorithm 3: Multiply a vector for a matrix $A \in Subset\left(GL_n(\mathbb{F}_q)\right)$

Require: $C_{i-1,j} \in \mathbb{F}_q$, $i = 2, \dots, n$ and $j = 0,1, \dots, i-2$.
Require: Monic polynomials $v_i(x) = v_{i,0} + v_{i,1}x + \cdots + v_{i,n-1}x^{n-i} + x^{n-i+1} \in \mathbb{F}_q[x]$, $v_{i,0} \neq 0, 1 \leq i \leq n$
Require: $l_i = 0,1, \dots, e_i - 1$ where e_i is the order of $v_i(x)$
Require: The vector $\vec{a} = (a_0, a_1, \cdots, a_{n-1})$
Ensure: $\vec{a} \cdot A$

for $i = n$ **down to** 2 **do:**
  $\hat{a}_0 = a_0 + C_{i,0} \cdot a_{i-1}$
  $\hat{a}_1 = a_1 + C_{i,1} \cdot a_{i-1}$
  ⋮
  $\hat{a}_{i-2} = a_{i-2} + C_{i,i-2} \cdot a_{i-1}$
  $a(x) = a_{i-1} + a_ix + \cdots + a_{n-1}x^{n-i}$
  $(\hat{a}_{i-1} + \cdots + \hat{a}_{n-1}x^{n-1}) = \big(a(x)\big)(x^{l_i}) \; mod \; v_i(x)$
  $(a_0, a_1, \dots., a_{n-1}) = (\hat{a}_0, \hat{a}_1, \dots, \hat{a}_{n-1})$
end for

$a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$
$(\hat{a}_0 + \cdots + \hat{a}_{n-1}x^{n-1}) = \big(a(x)\big)(x^{l_1}) \; mod \; v_1(x)$
return $(\hat{a}_0, \hat{a}_1, \dots, \hat{a}_{n-1})$

Algorithm 4: Multiply a vector for the matrix A^{-1}

Require: $C_{i-1,j} \in \mathbb{F}_q$, $i = 2, \dots, n$ and $j = 0,1, \dots, i-2$.
Require: Monic polynomials $v_i(x) = v_{i,0} + v_{i,1}x + \cdots + v_{i,n-1}x^{n-i} + x^{n-i+1} \in \mathbb{F}_q[x]$, $v_{i,0} \neq 0, 1 \leq i \leq n$
Require: $l_i = 0,1, \dots, e_i - 1$ where e_i is the order of $v_i(x)$
Require: The vector $\vec{a} = (a_0, a_1, \cdots, a_{n-1})$
Ensure: $\vec{a} \cdot A^{-1}$

$a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$
$(\hat{a}_0 + \cdots + \hat{a}_{n-1}x^{n-1}) = \big(a(x)\big)(x^{-l_1}) \; mod \; v_1(x)$
$(a_0, a_1, \dots., a_{n-1}) = (\hat{a}_0, \hat{a}_1, \dots, \hat{a}_{n-1})$

for $i = 2$ **to** n **do:**
  $\hat{a}_0 = a_0 - C_{i,0} \cdot a_{i-1}$
  $\hat{a}_1 = a_1 - C_{i,1} \cdot a_{i-1}$
  ⋮
  $\hat{a}_{i-2} = a_{i-2} + C_{i,i-2} \cdot a_{i-1}$
  $a(x) = a_{i-1} + a_ix + \cdots + a_{n-1}x^{n-i}$
  $(\hat{a}_{i-1} + \cdots + \hat{a}_{n-1}x^{n-1}) = \big(a(x)\big)(x^{-l_i}) \; mod \; v_i(x)$
  $(a_0, a_1, \dots., a_{n-1}) = (\hat{a}_0, \hat{a}_1, \dots, \hat{a}_{n-1})$
end for

return $(a_0, a_1, \dots., a_{n-1})$

REFERENCES

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C.: "Introduction to Algorithms". MIT Press, Massachusetts, 2022.

[2] Alman, J. and Williams, V. V.: "A refined laser method and faster matrix multiplication". In Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 522–539, 2021.

[3] Li, Y.-X., Li, D.-X., and Wu, C.-K.: "How to generate a random nonsingular, matrix in McEliece's public-key cryptosystem". In Proceedings Singapore ICCS/ISITA92, pp. 268–269, IEEE, 1992.

[4] Randall, D.: "Efficient generation of random nonsingular matrices". Random Structures & Algorithms, Vol. 4(1), pp. 111–118, 1993.

[5] Freyre, P., Díaz, N. and Morgado, E.: "Fast algorithm for the multiplication of a row vector by a randomly selected matrix A". Journal of Discrete Mathematical Sciences and Cryptography, Vol. 12(5), pp. 533–549, 2009.

[6] Murray, S.H.: "The Schereier-Sims algorithm". Essay submitted to the Department of

Mathematics of the Australian National University, 2003.

[7] Holt, D.F., Eick, B. and O'Brien, E.A.: "Handbook of computational group theory". CRC Press, 2005.

[8] Cannon, J. "A computational toolkit for finite permutation groups". In Proceedings of the Rutgers Group Theory Year, Vol. 1984, pp. 1–18, 1983.

[9] Green, J. A.: "Sets and groups: A first course in algebra". Springer, 1988.

[10] Peterson, W.W. and Weldon, E.J.: "Error-correcting codes". MIT Press, Massachusetts, 1972.

[11] Harvey, D. and Der Hoeven, J.: "Faster polynomial multiplication over finite fields using cyclotomic coefficient rings". Journal of Complexity, Vol. 54, pp. 101404, 2019.

[12] Cantor, D.G. and Kaltofen, E.: "On fast multiplication of polynomials over arbitrary algebras". Acta Informatica, Vol. 28(7), pp. 693–701, 1991.

[13] Brent, R.P. and Zimmermann, P.: "Modern Computer Arithmetic". Cambridge University Press, 2010.

[14] Althoen, S.C., Mclaughlin, R.: "Gauss-jordan reduction: A brief history". The American Mathematical Monthly, Vol. 94(2), pp. 130–142, 1987.

[15] Press, W.H., Teukolsky, S.A., Vetterling W.T. and Flannery, B.P.: "Numerical recipes: the art of scientific computing". Cambridge University Press, 1992.

[16] Krishnamoorthy, A. and Menon, D.: "Matrix inversion using Cholesky decomposition". In 2013 Signal Processing: Algorithms, Architectures, Arrangements and Applications, pp. 70–72, IEEE, 2013.

[17] Vajargah, B.F.: "A way to obtain Monte Carlo matrix inversion with minimal error". Applied Mathematics and Computation, Vol. 191(1), pp. 225–233, 2007.

[18] Huang, Y. and McColl, W.: "Analytical inversion of general tridiagonal matrices". Journal of Physics A: Mathematical and General, Vol. 30(22), 1997.

[19] Ries, F., De Marco, T. and Guerrieri, R.: "Triangular matrix inversión on heterogeneous multicore systems". IEEE Transactions on parallel and distributed systems, Vol. 23(1), pp. 177 – 184, 2011.

[20] Strassen, V. *et. al.*: "Gaussian elimination is not optimal". Numerische Mathematik, Vol. 13(4), pp. 354 – 356, 1969.

[21] Coppersmith, D. and Winograd, S.: "On the asymptotic complexity of matrix multiplication". SIAM Journal on Computing, Vol. 11(3), pp. 472 – 492, 1982.

[22] Traub, J.F.: "Associated polynomials and uniform methods for the solution of linear problems". SIAM Review, Vol 8(3), pp. 277 – 301, 1966.

ABOUT THE AUTHORS

Pablo Freyre Arrozarena
Workplace: Institute of Cryptography. University of Havana.
Email: pfreyre@matcom.uh.cu
Education: Graduated of Mathematics in 1988. Receive his Doctor's degree in 1998.
Research interests: Currently works in the fields of symmetric cryptography and post-quantum cryptography.

Tên tác giả: **Pablo Freyre Arrozarena**

Cơ quan công tác: Viện mật mã Đại học Havana, Cuba

Email: pfreyre@matcom.uh.cu

Quá trình đào tạo: Tốt nghiệp chuyên ngành Toán năm 1988. Nhận bằng Tiến sĩ năm 1998.

Hướng nghiên cứu hiện nay: Mật mã đối xứng, mật mã hậu lượng tử.

Alejandro Freyre Echevarría
Workplace: Institute of Cryptography. University of Havana.
Email: freyreealejandro@gmail.com
Education: Graduated of Computer Sciences in 2020. Received his Master's degree in 2023.
Research interests: Currently works in the fields of symmetric cryptography, optimization and post-quantum cryptography.

Tên tác giả: **Alejandro Freyre Echevarría**

Cơ quan công tác: Viện mật mã Đại học Havana, Cuba

Email: freyreealejandro@gmail.com

Quá trình đào tạo: Nhận bằng Đại học chuyên ngành Khoa học Máy tính năm 2020. Nhận bằng Thạc sĩ năm 2023.

Hướng nghiên cứu hiện nay: Mật mã đối xứng, mật mã hậu lượng tử.

Ernesto Dominguez Fiallo
Workplace: Institute of Cryptography. University of Havana.
Education: Graduated of Mathematics in 2015. Received his Master's degree in 2019.
Research interest: Currently works in the field of post-quantum cryptography, specifically in code-based cryptography.

Tên tác giả: **Ernesto Dominguez Fiallo**

Cơ quan công tác: Viện mật mã Đại học Havana, Cuba

Email: sontn.mta@gmail.com

Quá trình đào tạo: Nhận bằng Đại học chuyên ngành Toán năm 2015. Nhận bằng Thạc sĩ năm 2019.

Hướng nghiên cứu hiện nay: mật mã hậu lượng tử, mật mã dựa trên mã hóa.

Ramses Rodríguez Aulet
Workplace: Institute of Cryptography. University of Havana.
Email: ramsesrusia@yahoo.com
Education: Graduated of Mathematics in 2017. Receive his Master's degree in 2020.
Research interest: Currently works in the fields of symmetric cryptography and post-quantum cryptography.

Tên tác giả: **Ramses Rodríguez Aulet**

Cơ quan công tác: Viện mật mã Đại học Havana, Cuba

Email: ramsesrusia@yahoo.com

Quá trình đào tạo: Nhận bằng Đại học chuyên ngành Toán năm 2017. Nhận bằng Thạc sĩ năm 2020.

Hướng nghiên cứu hiện nay: Mật mã đối xứng, mật mã hậu lượng tử.

Samir Alzugaray Vizcaino
Workplace: Institute of Cryptography. University of Havana.
Education: Graduated of Mathematics in 2016.
Research interest: Currently works in the fields of symmetric cryptography and post-quantum cryptography.

Tên tác giả: **Samir Alzugaray Vizcaino**

Cơ quan công tác: Viện mật mã Đại học Havana, Cuba

Quá trình đào tạo: Nhận bằng Đại học chuyên ngành Toán năm 2016.

Hướng nghiên cứu hiện nay: Mật mã đối xứng, mật mã hậu lượng tử.