Algebraic Cryptanalysis of McEliece Variants with Compact Keys

Jean-Charles Faugère[1], Ayoub Otmani[2,3], Ludovic Perret[1], and Jean-Pierre Tillich[2]

[1] SALSA Project - INRIA (Centre Paris-Rocquencourt)
UPMC, Univ Paris 06 - CNRS, UMR 7606, LIP6
104, avenue du Président Kennedy 75016 Paris, France
jean-charles.faugere@inria.fr, ludovic.perret@lip6.fr
[2] SECRET Project - INRIA Rocquencourt
Domaine de Voluceau, B.P. 105 78153 Le Chesnay Cedex - France
ayoub.otmani@inria.fr, jean-pierre.tillich@inria.fr
[3] GREYC - Université de Caen - Ensicaen
Boulevard Maréchal Juin, 14050 Caen Cedex, France.

Abstract. In this paper we propose a new approach to investigate the security of the McEliece cryptosystem. We recall that this cryposystem relies on the use of error-correcting codes. Since its invention thirty years ago, no efficient attack had been devised that managed to recover the private key. We prove that the private key of the cryptosystem satisfies a system of bi-homogeneous polynomial equations. This property is due to the particular class of codes considered which are alternant codes. We have used these highly structured algebraic equations to mount an efficient key-recovery attack against two recent variants of the McEliece cryptosystems that aim at reducing public key sizes. These two compact variants of McEliece managed to propose keys with less than 20,000 bits. To do so, they proposed to use quasi-cyclic or dyadic structures. An implementation of our algebraic attack in the computer algebra system Magma allows to find the secret-key in a negligible time (less than one second) for almost all the proposed challenges. For instance, a private key designed for a 256-bit security has been found in 0.06 seconds with about $2^{17.8}$ operations.

Keywords : public-key cryptography, McEliece cryptosystem, algebraic cryptanalysis.

1 Introduction

Alternative cryptography. Despite the fact that several hard problems have been proposed as a foundation for public-key primitives, those effectively used are essentially classical problems coming from number theory: integer factorization (e.g. in RSA) and discrete logarithm (e.g. in Diffie-Hellman key-exchange). It is well-known that, although polynomial-time algorithms for those problems have not yet been found, they are not safe from a theoretic breakthrough that would endanger the security of the corresponding schemes. For instance, the emergence of a new computer model such as quantum computers would make schemes based on these classical number theory problems totally insecure.

The lack of diversity in public key cryptography has been identified as a major concern in the field of information security [27]. A good illustration of the potential damage of such lack of diversity is hash zoo. The portfolio of hash functions used so far in practice was mainly restricted to the same type of functions which are now almost all broken. Although the American National

Institute of Standards and Technology (NIST) issued an international call[4] to design a new standard hash function, the cryptography community will remain in a fuzzy situation until 2012 (date of final decision).

One of the main issues in public key cryptography is to identify hard problems that are not based on the classical ones coming from number theory. However, few emerged until now as a viable alternative. As pointed in [2], promising candidates include: the problem of solving multivariate equations over a finite field, the problem of finding a short vector in a lattice and the problem of decoding a linear code. Those problems known for being NP-hard are not concerned with the quantum computer threat.

McEliece cryptosystem. Among those problems, code-based cryptosystems seem to offer the most promising alternative. McEliece public key cryptosystem [24] is one of the oldest public-key cryptosystems. Its security relies on the difficulty of decoding a linear code. The main advantage of this system is to have very fast encryption and decryption functions. Depending on how the parameters are chosen for a fixed security level, this cryptosystem is about five times faster for encryption and about 10 to 100 times faster for decryption than RSA [10]. Furthermore, it has withstood many attacking attempts. After more than thirty years now, it still belongs to the very few public key cryptosystems which remain unbroken.

Following McEliece's pioneering work, several different public key cryptosystems based on the intractability of decoding a linear code have been proposed [28, 19, 31, 22, 7, 6, 4, 3, 5, 26]. The original McEliece cryptosystem relies on Goppa codes whereas its variants suggested to use different codes. The Sidelnikov system [31] used Reed-Muller codes, the Janwa-Moreno system proposed to take algebraic geometric codes [22] and the Gabidulin-Paramonov-Tretjakov (GPT) cryptosystem considered Gabidulin codes devised for the rank-metric. LDPC codes have also been repeatedly suggested for this use. Niederreiter is the first in [28] to bring in a significant modification of the McEliece system. However his suggestion to use Generalized Reed-Solomon codes turned out to be an insecure solution [32]. Many of these schemes were broken [32, 21, 25, 29, 17, 30, 34]. All these attacks result in a total break of the system (the secret key, or an equivalent secret key is recovered from the knowledge of the public key). However, the original McEliece remains unbroken. The fact that the best known attacks are still exponential speaks for itself [33, 8, 9, 18].

Despite its impressive resistance against a variety of attacks and its fast encryption and decryption, McEliece cryptosystem has not stood up to RSA for practical applications. This is most likely due to the large size of the public key which is between several hundred thousand and several million bits. To overcome this limitation, a new trend initiated in [20] manages to decrease the key size by choosing the public matrix defining the code in a particular form; for instance with a quasi-cyclic structure [20]. This enables to decrease significantly the key size. The same idea was used in [4] with LDPC codes. Both schemes were broken in [29]. It should be noted that both proposals did not use the Goppa codes of the McEliece cryptosystem, and the attacks have no impact on its security.

This work was then followed by two independent proposals [5, 26] that are based on the same kind of idea of using quasi-cyclic [5] or dyadic structure [26]. Both use the same type of codes called the alternant code family which contains Goppa codes. Actually the codes used in [26] are Goppa codes. This approach is quite attractive because it results in a drastic improvement of the public key size. In [5], the size ranges between 8000 and 20000 bits, whereas it lies between 4000 and 20000 bits in [26]. Until now, these new proposals seem to be immune against the attack suggested in [29].

[4] `http://csrc.nist.gov/groups/ST/hash/sha-3/index.html`.

Our contribution. In this paper we show that both schemes have a serious flaw that can be exploited to recover the private keys. We present an *algebraic cryptanalysis* of the quasi-cyclic and dyadic schemes [5, 26]. Algebraic cryptanalysis is a general framework that permits to assess the security of theoretically all cryptographic schemes. So far, such type of attacks has been applied successfully against several multivariate schemes and stream ciphers. To our knowledge, it is the first time that such an approach is used against code-based cryptosystems. The basic principle of this cryptanalysis is to associate to a cryptographic primitive a set of algebraic equations. The system of equations is constructed in such a way as to have a correspondence between the solutions of this system, and a secret information of the cryptographic primitive (for instance the secret key of an encryption scheme). In McEliece, the algebraic system that we have to solve for recovering the secret-key has the following very specific structure:

$$\left\{ g_{i,0}Y_0X_0^j + \cdots + g_{i,n-1}Y_{n-1}X_{n-1}^j = 0 \;\middle|\; i \in \{0,\ldots,k-1\}, j \in \{0,\ldots,r-1\} \right\} \tag{1}$$

where the unknowns are the X_i's and the Y_i's and the $g_{i,j}$'s are known coefficients (with $0 \leq i \leq k-1, 0 \leq j \leq n-1$) that belong to a certain field $\mathbb{F}_q$ with $q = 2^s$. We look for solutions of this system in a certain extension field $\mathbb{F}_{q^m}$. Here k is an integer which is at least equal to $n - rm$. By denoting $\mathbf{X} \stackrel{\text{def}}{=} (\mathbf{X_0}, \ldots, \mathbf{X_{n-1}})$ and $\mathbf{Y} \stackrel{\text{def}}{=} (\mathbf{Y_0}, \ldots, \mathbf{Y_{n-1}})$ we will refer to such an algebraic system by $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}, \mathbf{Y})$. The total number of equations is rk. The number of unknowns $2n$ and the maximum degree $r-1$ of the equations can be extremely high when cryptographic parameters are considered (e.g. $n = 1024$ and $r - 1 = 49$). Thus it is not clear whether an algebraic attack can be mounted efficiently in general.

However, in the case of the tweaked McEliece schemes what we study here we have either quasi-cyclic or dyadic [5, 26], it turns out that is possible to make use of this structure in order
(i) to reduce considerably the number of unknowns in the algebraic system (5),
(ii) moreover, it also appears that by using only the linear equations involving the Y_i's gives a linear space of solutions which is of small dimension.

We will explain in Section 4 and Section 5 respectively how to solve the underlying algebraic systems. We will see how the public-key structure (quasi-cyclic, or dyadic) impacts on the difficulty of solving the algebraic system (5). In particular, the structure induces an imbalance between the X and Y variables. From a practical point of view, we have been able for most of the parameters proposed in [5, 26] to recover the secret-key via Gröbner bases computations in a time ranging from few seconds (typically for [5]) to several hours ([26]). Before that, we will briefly recall in the next section the McEliece scheme and explain how we derive the algebraic system (5).

2 McEliece Public-Key Cryptosystem

We recall here how the McEliece public-key cryptosystem is defined.

Secret key: the triplet $(\boldsymbol{S}, \boldsymbol{G_s}, \boldsymbol{P})$ of matrices defined over a finite field $\mathbb{F}_q$ over q elements, with q being a power of two, that is $q = 2^s$. $\boldsymbol{G_s}$ is a full rank matrix of size $k \times n$, with $k < n$, $\boldsymbol{S}$ is of size $k \times k$ and is invertible, and $\boldsymbol{P}$ is permutation matrix of size $n \times n$. Moreover $\boldsymbol{G_s}$ defines a code (which is the set of all possible $\boldsymbol{uG_s}$ with $\boldsymbol{u}$ ranging over $\mathbb{F}_q^k$) which has a decoding algorithm which can correct in polynomial time a set of errors of weight at most t. This means that it can recover in polynomial time $\boldsymbol{u}$ from the knowledge of $\boldsymbol{uG_s} + \boldsymbol{e}$ for all possible $\boldsymbol{e} \in \mathbb{F}_q^n$ of Hamming weight at most t.

Public key: the matrix product $\boldsymbol{G} = \boldsymbol{SG_sP}$.

Encryption: A plaintext $\boldsymbol{u} \in \mathbb{F}_q^k$ is encrypted by choosing a random vector $\boldsymbol{e}$ in $\mathbb{F}_q^n$ of weight at most t. The corresponding ciphertext is $\boldsymbol{c} = \boldsymbol{uG} + \boldsymbol{e}$.

Decryption: $\boldsymbol{c}' = \boldsymbol{cP}^{-1}$ is computed from the ciphertext $\boldsymbol{c}$. Notice that $\boldsymbol{c}' = (\boldsymbol{uSG_sP}+\boldsymbol{e})\boldsymbol{P}^{-1} = \boldsymbol{uSG_s} + \boldsymbol{eP}^{-1}$ and that $\boldsymbol{eP}^{-1}$ is of Hamming weight at most t. Therefore the aforementioned decoding algorithm can recover in polynomial time $\boldsymbol{uS}$. This vector is multiplied by $\boldsymbol{S}^{-1}$ to obtain the plaintext $\boldsymbol{u}$.

This describes the general scheme suggested by McEliece. From now on, we will say that $\boldsymbol{G}$ is the *public generator matrix* and that the vector space $\mathscr{C}$ spanned by its rows is the *public code*, i.e.

$$\mathscr{C} \stackrel{\text{def}}{=} \{\boldsymbol{uG} | \boldsymbol{u} \in \mathbb{F}_q^k\}.$$

What is generally referred to as the McEliece cryptosystem is this scheme together with a particular choice of the code, which consists in taking a binary Goppa code. This class of codes belongs to a more general class of codes, namely the alternant code family ([23, Chap. 12, p. 365]). The main feature of this last class of codes is the fact that they can be decoded in polynomial time.

3 Algebraic Approach

3.1 Setting up the algebraic system

In this part, we explain more precisely how we construct the algebraic system described in (5). As explained in the previous section, the McEliece cryptosystem relies on Goppa codes which belong to the class of *alternant codes* and inherit an efficient decoding algorithm from this. We will describe this class of codes in more details since both cryptosystems that we cryptanalyze use codes from this class. It is convenient to describe this class through a *parity-check matrix*. This is an $r \times n$ matrix $\boldsymbol{H}$ defined over an extension $\mathbb{F}_{q^m}$ of the field over which the code is defined, which is such that

$$\{\boldsymbol{uG_s} \mid \boldsymbol{u} \in \mathbb{F}_q^k\} = \{\boldsymbol{c} \in \mathbb{F}_q^n \mid \boldsymbol{Hc}^T = 0\}. \tag{2}$$

r satisfies in this case the condition $r \geq \frac{n-k}{m}$.

In the case of alternant codes, there exists a parity-check matrix with a very special form related to Vandermonde matrices. More precisely there exist two vectors $\boldsymbol{x} = (x_0, \ldots, x_{n-1})$ and $\boldsymbol{y} = (y_0, \ldots, y_{n-1})$ in $\mathbb{F}_{q^m}^n$ such that $\boldsymbol{V}_r(\boldsymbol{x}, \boldsymbol{y})$ is a parity-check matrix , with

$$\boldsymbol{V}_r(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \begin{pmatrix} y_0 & \cdots & y_{n-1} \\ y_0 x_0 & \cdots & y_{n-1} x_{n-1} \\ \vdots & & \vdots \\ y_0 x_0^{r-1} & \cdots & y_{n-1} x_{n-1}^{r-1} \end{pmatrix}. \tag{3}$$

We use the following notation in what follows

Definition 1. *The alternant code of order r over $\mathbb{F}_q$ associated to $\boldsymbol{x} = (x_0, \ldots, x_{n-1}) \in \mathbb{F}_{q^m}^n$ and $\boldsymbol{y} = (y_0, \ldots, y_{n-1}) \in \mathbb{F}_{q^m}^n$ is denoted by $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ and is defined by*

$$\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) = \{\boldsymbol{c} \in \mathbb{F}_q^n | \boldsymbol{V}_r(\boldsymbol{x}, \boldsymbol{y})\boldsymbol{c}^T = 0\}.$$

It should be noted that the public code in the McEliece scheme is also an alternant code. We denote here by the public code, the set of vectors of the form

$$\{\boldsymbol{uG} \mid \boldsymbol{u} \in \mathbb{F}_q^k\} = \{\boldsymbol{cSG_sP} \mid \boldsymbol{c} \in \mathbb{F}_q^k\}.$$

This is simple consequence of the fact that the set $\{\boldsymbol{uSG_sP} \mid \boldsymbol{u} \in \mathbb{F}_q^k\}$ is obtained from the secret code $\{\boldsymbol{uG_s} \mid \boldsymbol{u} \in \mathbb{F}_q^k\}$ by permuting coordinates in it with the help of $\boldsymbol{P}$, since multiplying by an invertible matrix $\boldsymbol{S}$ of size $k \times k$ leaves the code globally invariant.

The key feature of an alternant code is the following fact

Fact 1. *There exists a polynomial time algorithm decoding all errors of Hamming weight at most an alternant code once a parity-check matrix $\boldsymbol{H}$ of the form $\boldsymbol{H} = \boldsymbol{V}_r(\boldsymbol{x}, \boldsymbol{y})$ is given for it.*

In other words, it is possible to break the McEliece scheme, if it is possible to find $\boldsymbol{x}^*$ and $\boldsymbol{y}^*$ in $\mathbb{F}_{q^m}^n$ such that

$$\{x\boldsymbol{G} | x \in \mathbb{F}_q\} = \{y \in \mathbb{F}_q | \boldsymbol{V}_r(\boldsymbol{x^*}, \boldsymbol{y^*})y^T = 0\}. \tag{4}$$

From the knowledge of this matrix $\boldsymbol{V}_r(\boldsymbol{x^*}, \boldsymbol{y^*})$, it is possible to decode the public code, therefore it is possible to recover $\boldsymbol{u}$ from $\boldsymbol{uG} + \boldsymbol{e}$.

Finding such a matrix $\boldsymbol{V}_r(\boldsymbol{x^*}, \boldsymbol{y^*})$ clearly amounts to find a matrix $\boldsymbol{V}_r(\boldsymbol{x^*}, \boldsymbol{y^*})$ such that

$$\boldsymbol{V}_r(\boldsymbol{x^*}, \boldsymbol{y^*})\boldsymbol{G}^T = \boldsymbol{0}.$$

By bringing in $2n$ variables $X_0, \ldots, X_{n-1}$ and $Y_0, \ldots, Y_{n-1}$ where X_i corresponds to x_i^* and Y_i to y_i^* we see that this is equivalent to solve the following system:

$$\left\{ g_{i,0}Y_0X_0^j + \cdots + g_{i,n-1}Y_{n-1}X_{n-1}^j = 0 \;\middle|\; i \in \{0, \ldots, k-1\}, j \in \{0, \ldots, r-1\} \right\} \tag{5}$$

where the $g_{i,j}$'s are the entries of the known matrix $\boldsymbol{G}$ with $0 \leq i \leq k-1$ and $0 \leq j \leq r-1$.

The cryptosystems proposed in [5, 26] follow the McEliece scheme [24] with the additional goal to design a public-key cryptosystem with very small key sizes. They both require to identify alternant codes having a property that allows matrices to be represented by very few rows. In the case of [5] circulant matrices are chosen whereas the scheme [26] focuses on dyadic matrices. These two families have in common the fact the matrices are completely described from the first row. The public generator matrix $\boldsymbol{G}$ in these schemes is a block matrix where each block is circulant in [5] and dyadic in [26].

We shall see that the algebraic approach previously described leads to a key-recovery in nearly all the parameters proposed in both schemes. The crucial point that makes the attack possible is the fact that we have a system with much less unknowns than in the case of the McEliece cryptosystem. This is due to both the particular structure of the matrices and their block form that describe the public alternant codes.

We finally end this section with a simple remark which explains that we can basically set one of the Y_i's and two values of the X_i's to an arbitrary value in the algebraic system (5).

Proposition 1 ([23, Chap. 10, p. 305]). *Let $X_0, X_1, \ldots, X_{n-1}, Y_0, \ldots, Y_{n-1}$ be a solution of (5) and $a \neq 0, b, c$ be elements of $\mathbb{F}_{q^m}$. Then $aX_0 + b, aX_1 + b, \ldots, aX_{n-1} + b, cY_0, \ldots, cY_{n-1}$ is also a solution of (5).*

3.2 Solving the Algebraic System $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X},\mathbf{Y})$

In this part, we describe a general technique to solve the algebraic system $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X},\mathbf{Y})$ using Gröbner bases techniques [11–14]. Although the particular characteristics of the cryptosystems [5, 26] studied here will further influence the shape of $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X},\mathbf{Y})$ (number of variables, number of equations, ...), we have designed a special strategy for taking advantage as much as possible of the intrinsic structure.

We have made an implementation of the strategy described here. We will present the experimental results, as well as the improvements which are possible due to the quasi-cyclic and dyadic structures, in Section 4 (quasi-cyclic case) and Section 5 (dyadic case).

As a first general remark, it is readily seen that $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X},\mathbf{Y})$ is highly structured. For instance, it is very sparse as the only monomials occuring in the system are of the form $Y_i X_i^j$, with $0 \leq i \leq k-1$ and $0 \leq j \leq r-1$. It can also be noticed that each block of k equations is *bi-homogeneous*, i.e. homogeneous if the variables of $\mathbf{X}$ (resp. $\mathbf{Y}$) are considered alone. Note that such structure already appears in the cryptanalysis of the MinRank problem [15].

Due to the particular structure of the system, it makes sense to design a specific strategy for solving $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X},\mathbf{Y})$. A simple way for solving this system would consist in generating the equations and try to solve it directly with a generic algorithm (for instance, the Gröbner basis algorithm available in the Magma computer algebra software). This approach fails for most challenges proposed in [5, 26]. However, it is interesting to remark that this direct approach has been mounted in practice for one challenge, namely A_{20}, of [5]. We will not provide much details on this. We only mention that it takes 24 hours of computation using a negligible amount of memory. As a comparaison, the improved strategy that we will describe now permits to solve (almost) all the challenges of [5, 26] in few seconds (using also a negligible amount of memory).

The first fundalemental remark is that there are k linear equations in the n variables of the block $\mathbf{Y}$ in $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X},\mathbf{Y})$. This implies that all the variables of the block $\mathbf{Y}$ can be expressed in terms of $d \geq n-k$ variables. From now on, we will always assume that the variables of the block $\mathbf{Y}'$ only refer to these d *free* variables. The first step is then to rewrite the system (5) only in function of the variables of $\mathbf{X}$ and $\mathbf{Y}'$, i.e., the variables of $\mathbf{Y} \setminus \mathbf{Y}'$ are substituted by linear combinations involving only variables of $\mathbf{Y}'$. In the particular cases of [5, 26], the quasi-cyclic and dyadic structures provide additional linear equations in the variables of $\mathbf{X}$ and $\mathbf{Y}'$ which can be also used to rewrite/clean the system. In the sequel, we will denote by $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}',\mathbf{Y}')$ the system obtained from $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X},\mathbf{Y})$ by removing all the linear equations in $\mathbf{X}$ and $\mathbf{Y}$.

The second crucial point is that as soon as the the projection of the solutions on the variables $\mathbf{Y}'$ are known, then the solutions on $\mathbf{X}$ can be efficiently found. Indeed, when the variables of $\mathbf{Y}'$ are fixed the system (5) simplifies to:

$$\left\{ g'_{i,0} X_0^j + \cdots + g'_{i,n-1} X_{n-1}^j = 0 \;\middle|\; i \in \{0,\ldots,k-1\}, j \in \{0,\ldots,r-1\} \right\}.$$

This system is readily solved with the help of the following trick. We first keep only the equations in this system which correspond to powers of the X_i's which are powers of two. In other words we consider only the equations of the form

$$g'_{i,0} X_0^{2^j} + \cdots + g'_{i,n-1} X_{n-1}^{2^j} = 0$$

for j in $\{0,\ldots,\lfloor \log_2(r-1) \rfloor\}$ and i in $\{0,\ldots,k-1\}$. Then, notice that the Frobenius map which associates to an x in $\mathbb{F}_{q^m} = \mathbb{F}_{2^{sm}}$ its 2^j-th power is an $\mathbb{F}_2$ linear transformation. This means that if we decompose x and $x' \stackrel{\text{def}}{=} x^{2^j}$ in a basis of $\mathbb{F}_2^{sm}$ as $x = (x_1,\ldots,x_{sm})$ and $x' = (x'_1,\ldots,x'_{sm})$,

then there exists a binary matrix M_j of size $sm \times sm$ such that $(x'_1, \ldots, x'_{sm}) = M_j(x_1, \ldots, x_{sm})$. By expressing all the X_i's in this basis we obtain therefore a linear system over $\mathbb{F}_2$ with smn unknowns and $sm\lfloor \log_2(r-1) \rfloor k$ equations. Since for all practical choices of the parameters we have that $\lfloor \log_2(r-1) \rfloor k > n$, this system permits (very likely) to reveal the X_i's.

It is worth to mention that, for the cryptosystems considered in this paper [5, 26], the number of free variables d in $\mathbf{Y}'$ can be rather small rather (typically 1 or 2 for some challenges). We can then perform an exhaustive search on these variables and recover $\mathbf{X}$ by solving a linear system as explained above. For some of the challenges proposed in [5, 26], this already leads to an attack which is practical, or almost practical. We will present below an even more efficient strategy to recover $\mathbf{Y}'$.

The most difficult part of the computation is to find a projection of the solutions with respect to the variables of the block $\mathbf{Y}'$. More formally, let $\mathcal{I}$ be the ideal generated by $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}', \mathbf{Y}')$ and $\mathcal{V}$ be the corresponding variety (i.e. the set of solutions). As already explained, the goal is to compute the projection of $\mathcal{V}$, denoted of $\mathcal{V}'$, on the variables of $\mathbf{Y}'$. This can done by computing a Gröbner basis (w.r.t. a degree order) G_{deg} of $\mathcal{I} \cap \mathbb{F}_{q^m}[\mathbf{Y}']$. This is a classical problem in computer algebra which can be solved by using standard elimination techniques (for instance see [1, Chap. 2.3, p. 69] or [12, Chap. 3, p. 112]. In the appendix, we briefly recall basic facts about elimination theory.

To be sure that the variety $\mathcal{V}'$ associated to $\mathcal{I} \cap \mathbb{F}_{q^m}[\mathbf{Y}']$ has few solutions, we have to remove parasite solutions (corresponding to $X_i = X_j$ or to $Y_j = 0$). A classical way to do that is to introduce new variables u_{ij} and v_i and add to $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}', \mathbf{Y}')$ equations of the form:

$$u_{ij} \cdot (X_i - X_j) + 1 = \text{and } v_i \cdot Y_i + 1 = 0.$$

In practice, we have not added all theses equations; but only few of them (namely 4 or 5). The reason is that we do not want to add too many new variables. In addition, including few of the equations below already permits to remove trivial solutions. We also have to remove some degree of freedom in the algebraic system. To do so, we can fix randomly few variables of $\mathbf{X}$' (as explained in Proposition 1). It is important to notice that since we are removing component of high dimension the new system is indeed much faster to solve.

In our context, we have used a slightly modified version of F_4 [13] for computing a Gröbner basis G_{deg} of $\mathcal{I} \cap \mathbb{F}_{q^m}[\mathbf{Y}']$. Roughly speaking, the idea is to adapt the algorithm for performing the Gröbner basis computation in $\mathbb{F}_{q^m}[\mathbf{X}'][\mathbf{Y}']$, i.e. the set of polynomials in $\mathbf{Y}'$ whose coefficients are polynomials in $\mathbb{F}_{q^m}[\mathbf{X}']$. As for the usual F_4, we process degree by degree. However, we consider only the degree of the polynomials w.r.t. the variables of $\mathbf{X}'$. We stop the computation as soon as we have sufficiently many equations in $\mathbf{Y}'$ (in other words, as soon as we detect that $\mathcal{V}'$ has a finite number of solution, i.e. is of dimension zero).

To compute G_{deg}, we have not considered all the equations of $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}', \mathbf{Y}')$. This system being naturally over-defined, we can "safely" remove some equations. Typically, it makes sense to consider the smaller subset of equations such that $\mathcal{V}'$ is zero-dimensional and for which we can efficiently compute G_{degree}. In [5, 26], the system $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}', \mathbf{Y}')$ is always defined over a field of characteristic two. We have then considered the set of equations of $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}', \mathbf{Y}')$ whose degree in the variables of $\mathbf{X}'$ is a power of two. Hence, we obtain a quasi bi-linear system.

To summarize, the strategy is divided in three steps. We first remove all linear equations from $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}, \mathbf{Y})$ to obtain $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}', \mathbf{Y}')$. We then compute the projection $\mathcal{V}'$ of $\mathcal{V}$ on the variables of $\mathbf{Y}'$ by fixing few variables of $\mathbf{X}'$. To do so, it is sufficient to compute a Gröbner basis G_{deg} of $\mathcal{I} \cap \mathbb{F}_{q^m}[\mathbf{Y}']$. This can be done with the tweaked version of F_4 given in the Appendix. Once the components of the variety corresponding to $\mathbf{Y}'$ are obtained, we specialize such values in the full system $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}', \mathbf{Y}')$ and recover the variables $\mathbf{X}'$ by linearizing the system. Note

that if this linearization fails, we can alternatively compute a Gröbner basis of $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}', \mathbf{Y}')$ where the variables of $\mathbf{Y}'$ have been fixed. Although this system is non linear, it has very few monomials per equation. It is in some some sense quasi-linear and can be easily solved in practice. Below, we describe more precisely the procedure which allows to compute a Gröbner basis G_{deg} of $\mathcal{I} \cap \mathbb{F}_{q^m}[\mathbf{Y}']$. As already explained, this is the most difficult part.

Algorithm 1: ComputeProjection

Input : The system $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}', \mathbf{Y}')$
Output: A Gröbner basis G_{lex} of $\mathcal{I} \cap \mathbb{F}_{q^m}[\mathbf{Y}']$

Let F be the equations of degree $2^i, 1 \leq i \leq r-1$ of $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}', \mathbf{Y}')$
Let F' be the system obtained from F by fixing "randomly" some variables of $\mathbf{X}'$
Compute a Gröbner basis G_{deg} of $\mathcal{I} \cap \mathbb{F}_{q^m}[\mathbf{Y}']$ using the tweaked version of F_4
Return G_{deg}

Note that the number of variables fixed to obtain F' is specified in Proposition 1. The variety $\mathcal{V}'$ having few elements it is not difficult to recover this set from the knowledge of G_{deg}.

4 Algebraic Cryptanalysis of the Quasi-Cyclic Variant

4.1 Description of the scheme

The scheme presented in [5] suggests to use block matrices where each block is a circulant matrix. An $\ell \times \ell$ matrix $\boldsymbol{A}$ is circulant if there exists an ℓ-tuple $\boldsymbol{a} = (a_0, \ldots, a_{n_0-1})$ such that:

$$\boldsymbol{A} = \begin{pmatrix} a_0 & a_1 & \cdots & a_{\ell-1} \\ a_{\ell-1} & a_0 & \cdots & a_{\ell-2} \\ \vdots & \ddots & \ddots & \vdots \\ a_1 & \cdots & a_{\ell-1} & a_0 \end{pmatrix}.$$

It is clear that a circulant matrix is uniquely defined by its first row.

The public code $\mathscr{C}$ of the McEliece like scheme suggested in [5] depends on the following choices

1. $\mathscr{C}$ is defined on a field $\mathbb{F}_q = \mathbb{F}_{2^s}$ and also depends on a certain extension field $\mathbb{F}_{q^m}$.
2. Let ℓ be some divisor of $q^m - 1$, i.e. there exists N_0 such that $q^m - 1 = N_0\ell$.
3. The length n of the code is chosen to be a multiple of ℓ, $n = n_0\ell$ with $n_0 \leq N_0$;
4. A certain integer e is chosen in the range $\{0, \ldots, \ell - 1\}$.
5. A certain integer r is chosen such that $n > rm$ and such that rm is a multiple of ℓ. Let δ be such that $rm = \delta\ell$;
6. Let α be a primitive element of $\mathbb{F}_{q^m}$ and $\beta \stackrel{\mathrm{def}}{=} \alpha^{N_0}$. Note that β is of order ℓ: $\beta^\ell = \alpha^{N_0\ell} = \alpha^{q^m-1} = 1$.
7. Let $\boldsymbol{d} \stackrel{\mathrm{def}}{=} (d_0, \ldots, d_{n_0-1})$ be an n_0-tuple of integers belonging to $\{0, \ldots, \ell - 1\}$ and $\boldsymbol{w} \stackrel{\mathrm{def}}{=} (w_0, \ldots, w_{n_0-1})$ be an n_0-tuple of different integers in the range $\{0, \ldots, N_0 - 1\}$.
8. Let $\boldsymbol{\gamma} \stackrel{\mathrm{def}}{=} (\gamma_0, \ldots, \gamma_{n_0-1})$ be an n_0-tuple of non-zero elements of $\mathbb{F}_{q^m}$.

Although this is not explicitly stated in [5], it is readily checked from the description given there that the public code $\mathscr{C}$ is defined from a certain matrix $\boldsymbol{H}$ as

$$\mathscr{C} = \{\boldsymbol{c} \in \mathbb{F}_q^n | \boldsymbol{H}\boldsymbol{c}^T = 0\},$$

with $\boldsymbol{H}$ being a matrix over $\mathbb{F}_{q^m}$ of size $r \times n$ which can be decomposed in n_0 blocks of size $r \times \ell$ as : $\boldsymbol{H} = \left(\boldsymbol{H}^{(0)} \middle| \boldsymbol{H}^{(1)} \middle| \cdots \middle| \boldsymbol{H}^{(n_0-1)}\right)$ where the block $\boldsymbol{H}^{(b)} = (h_{i,j}^{(b)})_{\substack{0 \le i \le r-1 \\ 0 \le j \le \ell-1}}$ is given by

$$h_{i,j}^{(b)} = \gamma_b \beta^{(d_b+j)e} \left(\alpha^{w_b} \beta^{d_b+j}\right)^i. \tag{6}$$

From this description, it is now clear that the code $\mathscr{C}$ is an alternant code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ of order r associated to $\boldsymbol{x} = (x_0, \ldots, x_{n-1})$ and $\boldsymbol{y} = (y_0, \ldots, y_{n-1})$ which are given by

$$x_{b\ell+j} = \alpha^{w_b} \beta^{d_b+j} \tag{7}$$

$$y_{b\ell+j} = \gamma_b \beta^{(d_b+j)e}, \tag{8}$$

for j in $\{0, \ldots, \ell-1\}$.

It can be checked (see [5]) that $\mathscr{C}$ has a public generating matrix $\boldsymbol{G}$ which is block circulant of size $k \times n$ with k of the form $k = k_0\ell$ and k_0 is some integer satisfying $k_0 \ge n_0 - \delta$. By block circulant, we mean here that we can decompose the matrix $\boldsymbol{G}$ as $\boldsymbol{G} = (\boldsymbol{G}_{i,j})_{\substack{0 \le i \le k_0-1 \\ 0 \le j \le n_0-1}}$ in blocks $\boldsymbol{G}_{i,j}$, each block being a circulant matrix of size $\ell \times \ell$. This induces a significant reduction in the public key size, since in order to specify $\boldsymbol{G}$ we just have to give the first row of each $\boldsymbol{G}_{i,j}$. There is also some significant gain for the secret key size. Indeed, from Fact 1 it is sufficient to being able to recover $\boldsymbol{x}$ and $\boldsymbol{y}$ in order to decode $\mathscr{C}$. This can be done by storing only $\boldsymbol{w}, \boldsymbol{d}, \boldsymbol{\gamma}, e$.

4.2 Algebraic Attack

In this part, we present an algebraic attack against the quasi-cyclic variant proposed in [5]. This starts by trying to recover $\boldsymbol{x}$ and $\boldsymbol{y}$ by using that

$$\boldsymbol{H}\boldsymbol{G}^T = 0. \tag{9}$$

This is a simple consequence of the fact that by definition of $\mathscr{C}, \boldsymbol{H}$ and $\boldsymbol{G}$ we have

$$\mathscr{C} = \{\boldsymbol{c} \in \mathbb{F}_q^n | \boldsymbol{H}\boldsymbol{c}^T = 0\} = \{\boldsymbol{u}\boldsymbol{G} | \boldsymbol{u} \in \mathbb{F}_q^k\}.$$

We can set up from (9) an algebraic system of the form $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}, \mathbf{Y})$ since (9) implies that

$$\sum_{j=0}^{n-1} g_{i,j} y_j x_j^w = 0 \tag{10}$$

where the $g_{i,j}$ are the entries of $\boldsymbol{G}$, i is between 0 and $k-1$ and w is in the range $\{0, \ldots, r-1\}$.

This gives a system with $2n$ unknowns. We can obtain a huge reduction of the number of unknowns by using Equations (7) and (8) which induce some linear relations between the x_i's and the y_i's. From these two equations we deduce that

$$x_{b\ell+j} = x_{b\ell} \beta^j \tag{11}$$

$$y_{b\ell+j} = y_{b\ell} \beta^{je}, \tag{12}$$

for any j in $\{0, \ldots, \ell-1\}$ and j in $\{0, \ldots, n_0-1\}$. Notice now, that we may assume that

Assumption 2. *The secret integer e such that $0 \leq e \leq \ell - 1$ is assumed to be known.*

This can be done without loss of generality since in the cases considered in [5], e is small because it lies in the range $\{0, \ldots, \ell-1\}$ and ℓ is less than 100 for all parameter choices. We may therefore try by brute force all possible values for e.

This enables to simplify the description of the system $\mathcal{M}\mathrm{cE}_{k,n,r}(\mathbf{X}, \mathbf{Y})$. By setting up the unknown X_b for $x_{b\ell}$ and the unknown Y_b for $y_{b\ell}$ we obtain the following algebraic system.

Proposition 2. *Let $\boldsymbol{G} = (g_{i,j})_{\substack{0 \leq i \leq k-1 \\ 0 \leq j \leq n-1}}$ be the $k \times n$ public generator matrix with $k = k_0\ell$ and $n = n_0\ell$. For any $0 \leq w \leq r-1$ and any $0 \leq i \leq k-1$, the unknowns $X_0, \ldots, X_{n_0-1}$ and $Y_0, \ldots, Y_{n_0-1}$ should satisfy:*

$$\sum_{b=0}^{n_0-1} g'_{b,i,w} Y_b X_b^w = 0 \qquad \text{where} \quad g'_{b,i,w} \stackrel{def}{=} \sum_{j=0}^{\ell-1} g_{i,b\ell+j} \beta^{j(e+w)}. \tag{13}$$

Proof. We observe that

$$\begin{aligned}
\sum_{j=0}^{n-1} g_{i,j} y_j x_j^w &= \sum_{b=0}^{n_0-1} \sum_{j=0}^{\ell-1} g_{i,b\ell+j} y_{b\ell+j} x_{b\ell+j}^w \\
&= \sum_{b=0}^{n_0-1} \sum_{j=0}^{\ell-1} g_{i,b\ell+j} y_{b\ell} x_{b\ell}^w \beta^{je} \beta^{jw} \\
&= \sum_{b=0}^{n_0-1} \left(\sum_{j=0}^{\ell-1} g_{i,b\ell+j} \beta^{je} \beta^{jw} \right) y_{b\ell} x_{b\ell}^w
\end{aligned}$$

By setting X_b for $x_{b\ell}$ and Y_b for $y_{b\ell}$ we obtain the aforementioned system.

Thanks to Proposition 1, we know that we can fix two variables, say X_0 and X_1, and one variable Y_j, for instance Y_0, to almost arbitrary values. We can specialize these variables as long as $X_0 \neq X_1$ and $Y_0 \neq 0$. The total number of unknowns is actually $2(n_0 - 1) - 1$. However there are also many redundant equations in Proposition 2. This comes from the block circulant form of $\boldsymbol{G}$. From this form we know that

$$g_{i\ell+u,b\ell+j} = g_{i\ell,b\ell+((j-u) \mod \ell)} \tag{14}$$

for all u in $\{0, \ldots, \ell-1\}$, i in $\{0, \ldots, k_0 - 1\}$. Let us notice that for such an i and a u we have

$$\begin{aligned}
g'_{b,i\ell+u,w} &= \sum_{j=0}^{\ell-1} g_{i\ell+u,b\ell+j} \beta^{j(e+w)} \\
&= \sum_{j=0}^{\ell-1} g_{i\ell,b\ell+((j-u) \mod \ell)} \beta^{j(e+w)} \qquad (15) \\
&= \sum_{j=0}^{\ell-1} g_{i\ell,b\ell+j} \beta^{j(e+w)} \beta^{u(e+w)} \qquad (16) \\
&= g'_{b,i\ell,w} \beta^{u(e+w)}
\end{aligned}$$

Here we have used (14) in (15) and the fact that $\beta^{\ell(e+w)} = 1$ for obtaining Equation (16). In other words, the equations

$$\sum_{b=0}^{n_0-1} g'_{b,i\ell+u,w} Y_b X_b^w = 0$$

are all equivalent for a given i. This means that instead of having rk equations we have only $\frac{rk}{\ell} = k_0 r$ algebraic equations.

Proposition 3. *The system of polynomial equations defined in (13) consists of* $2(n_0 - 1) - 1$ *unknowns and* $rk/\ell = rk_0$ *equations.*

From now on, we will always assume that redudant equations have been removed, and the variables X_0, X_1, and Y_0 are fixed.

4.3 Experimental results

We will now present experimental results obtained when solving the system described in (13) using the strategy described in Section 3.2. The experimental results have been obtained with several Xeon bi-processor 3.2 Ghz, with 16 Gb of Ram. The instances of our problem have been generated using the Magma software. We used the Magma version 2.15 for our computations. The F_5 [14] algorithm has been implemented in C language in the FGb software. We used this implementation for computing the first Gröbner basis (i.e. which is used in Algorithm 3.2). All the other computations are performed under Magma including factorizing some univariate polynomials and computing Gröbner bases using the F_4 [13] algorithm.

Table 1. Cryptanalysis results for [5] ($m = 2$)

Challenge	q	ℓ	n_0	δ	Security [5]	Unkonwns	Equations	Time (Operations, Memory)
A_{16}	2^8	51	9	4	80	15	510	0.06 sec ($2^{18.9}$ op, 115 Meg)
B_{16}	2^8	51	10	4	90	17	612	0.03 sec ($2^{17.1}$ op, 116 Meg)
C_{16}	2^8	51	12	4	100	21	816	0.05 sec ($2^{16.2}$ op, 116 Meg)
D_{16}	2^8	51	15	5	120	27	1275	0.02 sec ($2^{14.7}$ op, 113 Meg)
A_{20}	2^{10}	75	6	3	80	9	337	0.05 sec ($2^{15.8}$ op, 115 Meg)
B_{20}	2^{10}	93	6	3	90	9	418	0.05 sec ($2^{17.1}$ op, 115 Meg)
C_{20}	2^{10}	93	8	5.67	110	13	697	0.02 sec ($2^{14.5}$ op, 115 Meg)
QC_{600}	2^8	255	15	4	600	27	6820	0.08 sec ($2^{16.6}$ op, 116 Meg)

The most important observation is that we have been able to solve all the challenges of [5] in a negligible time. We also proposed a challenge QC_{600} for showing the behaviour of our attack for high security levels.

5 Algebraic Cryptanalysis of the Dyadic Variant

5.1 Description of the scheme

The cryptosystem presented in [26] considers particular alternant codes called *quasi-dyadic* Goppa codes. Goppa codes form an important subclass of alternant codes. Goppa codes are

defined by means of a polynomial $G(X)$ of degree ℓ with coefficients in $\mathbb{F}_{q^m}$ and for which the sequence $\boldsymbol{x}$ is assumed not to contain any root of $G(X)$. The alternant code defined by the parity-check matrix $\boldsymbol{V}_\ell(\boldsymbol{x}, \boldsymbol{y})$ with $y_i = G(x_i)^{-1}$ is called a Goppa code over $\mathbb{F}_q$ and is denoted by $\mathscr{G}(\boldsymbol{x}, G)$. It has dimension $n - m\ell$ and minimum distance $d \geq \ell + 1$ [23, Chap. 12, p. 340]. In the special case where the roots $\boldsymbol{z} = (z_0, \ldots, z_{\ell-1})$ of $G(X)$ are distinct and all belong to $\mathbb{F}_{q^m}$ then $\mathscr{G}(\boldsymbol{x}, G)$ admits a parity-check matrix $\boldsymbol{C}(\boldsymbol{z}, \boldsymbol{x})$ in Cauchy form [23, p. 345].

The scheme in [26] considers a Goppa code that admits a parity-check matrix that is both a Cauchy Cauch matrix and a block matrix where each block is dyadic. An $\ell \times \ell$ matrix $\boldsymbol{\Delta} = (\Delta_{i,j})$ with $0 \leq i \leq \ell - 1$ and $0 \leq j \leq \ell - 1$ is *dyadic* if and only if $\Delta_{i,j} = h_{i \oplus j}$ where $\oplus$ is the bitwise exclusive-or on the binary representation of the indices and $\boldsymbol{h} = (h_0, \ldots, h_{\ell-1})$ is the first row of $\boldsymbol{\Delta}$. Let $\boldsymbol{h} = (h_0, \ldots, h_{N-1})$ be a vector of $\mathbb{F}_{q^m}^N$ with $\ell \leq N$. We denote by $\boldsymbol{\Delta}_\ell(\boldsymbol{h}) = (\Delta_{i,j})$ the $\ell \times N$ matrix such that $\Delta_{i,j} = h_{i \oplus j}$. One can easily observe that $\boldsymbol{\Delta}_\ell(\boldsymbol{h})$ is the juxtaposition of N_0 dyadic matrices of size $\ell \times \ell$ when $N = N_0 \ell$ for some integer N_0. Proposition 4 proved in [26, Theorem 2] characterizes dyadic Cauchy matrices.

Proposition 4. *A necessary and sufficient condition for $\boldsymbol{\Delta}_\ell(\boldsymbol{h})$ to be a Cauchy matrix $\boldsymbol{C}(\boldsymbol{z}, \boldsymbol{x})$ is that $\mathbb{F}_{q^m}$ is of characteristic 2 and for any i, j in $\{0, \ldots, N-1\}$ we have:*

$$\frac{1}{h_{i \oplus j}} = \frac{1}{h_j} + \frac{1}{h_i} + \frac{1}{h_0}. \tag{17}$$

Furthermore, for any $\theta \in \mathbb{F}_{q^m}$ and for any $z_i^* = 1/h_i + \theta$ and $x_j^* = 1/h_j + 1/h_0 + \theta$, the Cauchy matrix $\boldsymbol{C}(\boldsymbol{z}^*, \boldsymbol{x}^*)$ is equal to $\boldsymbol{\Delta}_\ell(\boldsymbol{h})$.

A detailed description of the key generation is given in Appendix (Section B). We only provide important facts that are useful for recovering the private key. Indeed, the public generator matrix $\boldsymbol{G}$ is a $k \times n$ block matrix where each block is an $\ell \times \ell$ dyadic matrix with ℓ being a power of 2. The entries of $\boldsymbol{G}$ belong to $\mathbb{F}_q$ and the integers k and n are chosen such that $n = n_0 \ell$ and $k = n - m\ell = \ell(n_0 - m)$ where n_0 is some integer and m defines the extension $\mathbb{F}_{q^m}$. The matrix $\boldsymbol{G}$ is obtained from a secret $\ell \times n$ block parity-check matrix $\boldsymbol{H} = \left(\boldsymbol{\Delta}_\ell(\boldsymbol{f}_0) | \cdots | \boldsymbol{\Delta}_\ell(\boldsymbol{f}_{n_0-1})\right)$ where each block $\boldsymbol{\Delta}_\ell(\boldsymbol{f}_j)$ is an $\ell \times \ell$ dyadic matrix and for any $0 \leq j \leq n_0 - 1$, $\boldsymbol{f}_j$ is a vector of $\mathbb{F}_{q^m}^\ell$ such that:

$$\boldsymbol{f}_j = \gamma_j \left(h_{\omega_j \ell \oplus d_j}, h_{(\omega_j \ell + 1) \oplus d_j}, \ldots, h_{((\omega_j + 1)\ell - 1) \oplus d_j}\right) \tag{18}$$

where $\boldsymbol{h} = (h_0, \ldots, h_{N-1})$ is a random vector of $\mathbb{F}_{q^m}^N$ that satisfies Equation (17) and such that $N = N_0 \ell$ for some integer $N_0 \gg n_0$. The integers ω_j, d_j are chosen such that $0 \leq \omega_j \leq N_0 - 1$ and $0 \leq d_j \leq \ell - 1$. The coefficients γ_j are non zero elements of $\mathbb{F}_{q^m}$. Note that the integers ω_j's are different. The secret key consists then of the vectors $\boldsymbol{h}$, $\boldsymbol{\omega} = (\omega_0, \ldots, \omega_{n_0-1})$, $\boldsymbol{d} = (d_0, \ldots, d_{n_0-1})$ and $\boldsymbol{\gamma} = (\gamma_0, \ldots, \gamma_{n_0-1})$.

5.2 Algebraic attack

We now make explicit the algebraic attack that leads to recover the private key. We first state an important result that shows that $\boldsymbol{G}$ defines actually an alternant code. The proof is in the Appendix (Section B).

Proposition 5. *The code defined by the public generator matrix $\boldsymbol{G}$ is an alternant code $\mathscr{A}_\ell(\boldsymbol{x}, \boldsymbol{y})$ where for any $0 \leq j \leq n_0 - 1$ and $0 \leq i, i' \leq \ell - 1$, we have the following equations:*

$$\begin{cases} y_{j\ell+i} & = y_{j\ell} \\ x_{j\ell+i} + x_{j\ell} & = x_i + x_0 \\ x_{j\ell+(i \oplus i')} & = x_{j\ell+i} + x_{j\ell+i'} + x_{j\ell} \end{cases} \tag{19}$$

The cryptanalysis of the system consists in defining n_0 unknowns $Y_0, \ldots, Y_{n_0-1}$ that play the role of the y_j's and n unknowns $X_0, \ldots, X_n$ that represent the x_j's. We know specify the system of equations that we obtain directly from Proposition 5.

Proposition 6. *For any w, j, i and i' such that $0 \leq w \leq \ell - 1$, $0 \leq j \leq n_0 - 1$ and $1 \leq i, i' \leq \ell - 1$, we have:*

$$\begin{cases} \displaystyle\sum_{j=0}^{n_0-1} Y_j \sum_{l=0}^{\ell-1} g_{i,j\ell+l} X_{j\ell+l}^w = 0 \\ \\ X_{j\ell+i} + X_{j\ell} + X_i + X_0 = 0 \\ \\ X_{j\ell+(i\oplus i')} + X_{j\ell+i} + X_{j\ell+i'} + X_{j\ell} = 0 \end{cases} \tag{20}$$

It is possible to simplify System (20) by observing, thanks to the third equation, that actually many variables X_i's can be expressed in function of some few variables, namely X_{2^j} with $0 \leq j \leq \log_2(\ell - 1)$ and X_b with $0 \leq b \leq n_0 - 1$.

Corollary 1. *For any $1 \leq i \leq \ell - 1$, if we write the binary decomposition of $i = \sum_{j=0}^{\log_2(\ell-1)} \eta_j 2^j$ then the following equation holds:*

$$X_i = X_0 + \sum_{j=0}^{\log_2(\ell-1)} \eta_j (X_{2^j} + X_0).$$

We are also able to provide the exact number of unknowns left after substitution and the effective number of equations after elimination of redundant equations.

Proposition 7. *The system (20) has $n_0 - 1$ unknowns Y_i and $n_0 - 2 + \log_2(\ell)$ unknowns X_i. Furthermore, it has $n_0 - m$ linear equations involving only the Y_i's, and $(\ell - 1)\ell(n_0 - m)$ polynomial equations involving the unknowns $Y_i X_i^w$ with $w > 1$.*

Proof. The number of variables Y_j is $(n_0 - 1)$ since we can choose $Y_0 = 1$. As for the variables X_j, we observe that they can all be expressed only in function of X_{2^j} and $X_{i\ell}$ with $0 \leq j \leq \log_2(\ell - 1)$ and $0 \leq i \leq n_0 - 1$. So the number of unknowns X_j is $\log_2(\ell - 1) + 1 + n_0 - 2$ since we can fix two different arbitrary values for two variables, say X_0 and X_ℓ (Proposition 1). Using the fact that $\log_2(\ell - 1) = \log_2(\ell) - 1$ since ℓ is a power of 2, we get the claimed number of unknowns. Furthermore, because of the dyadicity of $\boldsymbol{G}$, the equations obtained with $w = 0$ are identical when $\boldsymbol{g}$ describes all the rows of a dyadic block of $\boldsymbol{G}$. This does not appear when $w > 1$. So we have $k/\ell = n_0 - m$ linear equations that involve the Y_i's and $(\ell - 1)k = (\ell - 1)\ell(n_0 - m)$ polynomial equations that contain variables of the form $Y_i X_i^w$ where $w > 1$.

5.3 Experimental results

We now present in Table 2 the experimental results we obtained when we solve the system described in (20) using the strategy described in Section 3.2. As previously, the experimental results have been obtained with several Xeon bi-processor 3.2Ghz, with 16 Gb of Ram. The instances of our problem have been generated using the Magma software. We used the Magma version 2.15 for our computations. The F_5 [14] algorithm has been implemented in C language in the FGb software. We used this implementation for computing the first Gröbner basis (*i.e.* which is used in Algorithm 3.2). All the other computations are performed under Magma including factorizing some univariate polynomials and computing Gröbner bases using the F_4

algorithm [13]. Table 2 also shows the impact of the degree extension m. Indeed, computation times indicate that the solutions are easy to find if m is small. This phenomenon is directly related to the size of the solution space of the variables Y_i. We have seen in Section 3.2 that such variables satisfy a system of linear equations. From Proposition 7, the number of linear equations is $k_0 = n_0 - m$ whereas the number of unknowns Y_i is $n_0 - 1$. Thus the dimension of the vector space solution for the Y_i's is $m - 1$. We recall that once the Y_i are solved, we find the solutions for the X_i's by solving a system of linear equations. We also give in Table 2 new parameters (Dyadic_{256} and Dyadic_{512}) that are generated by "extrapolating[5]" challenges given in [26]. We observe that this solution space is manageable in practise as long as $m < 16$ because we did not succeed to find an efficient way to solve the challenges of [26] when $m = 16$.

Table 2. Cryptanalysis results for [26].

Challenge	q	m	ℓ	n_0	Security	Unkonwns	Equations	Time (Operations, Memory)
Table 2	2^2	8	64	56	128	115	193,584	1,776.3 sec ($2^{34.2}$ op, 360 Meg)
Table 2	2^4	4	64	32	128	67	112,924	0.50 sec ($2^{22.1}$ op, 118 Meg)
Table 2	2^8	2	64	12	128	27	40,330	0.03 sec ($2^{16.7}$ op, 35 Meg)
Table 3	2^8	2	64	10	102	23	32,264	0.03 sec ($2^{15.9}$ op, 113 Meg)
Table 3	2^8	2	128	6	136	16	65,028	0.02 sec ($2^{15.4}$ op, 113 Meg)
Table 3	2^8	2	256	4	168	13	130,562	0.11 sec ($2^{19.2}$ op, 113 Meg)
Table 5	2^8	2	128	4	80	12	32,514	0.06 sec ($2^{17.7}$ op, 35 Meg)
Table 5	2^8	2	128	5	112	14	48,771	0.02 sec ($2^{14.5}$ op, 35 Meg)
Table 5	2^8	2	128	6	128	16	65,028	0.01 sec ($2^{16.6}$ op, 35 Meg)
Table 5	2^8	2	256	5	192	15	195,843	0.05 sec ($2^{17.5}$ op, 35 Meg)
Table 5	2^8	2	256	6	256	17	261,124	0.06 sec ($2^{17.8}$ op, 35 Meg)
Dyadic_{256}	2^4	4	128	32	256	68	455,196	7.1 sec ($2^{26.1}$ op, 131 Meg)
Dyadic_{512}	2^8	2	512	6	512	18	1,046,532	0.15 sec ($2^{19.7}$ op, 38 Meg)

6 Conclusion

We described in this paper a new algebraic approach to assess the security of the McEliece cryptosystem. We would like to emphasize that our attach is structural, and could permit to recover the secret key. We showed that the private key of this scheme is a solution of a very structured system of bi-homogeneous polynomial equations in two sets of unknowns Y_i and X_i. The solutions belong to a finite field $\mathbb{F}_{q^m}$ whereas the coefficients of the system are in $\mathbb{F}_q$ for some known integers m and q. This system comes from the particular structure of alternant codes used in McEliece. Indeed, the Goppa codes as proposed in [24] form a subfamily of alternant codes. Furthermore, the system is composed of two parts of equations: one part that consists of linear equations that involve only the unknowns Y_i and a second part where the equations involve terms of the form $Y_i X_i^j$. Our analysis shows that if one is able to find the solutions for the Y_i's then the solutions of X_i can be usually recovered efficiently by solving a linear system. A direct example that we did not treat explicitly in this paper is the case of BCH codes. Such codes represent a subfamily of alternant codes for which it is known that a valid solution is

[5] We give in Table 2 the security we would have if the figures given in [26] were accurate. Note we do not know how they were calculated.

obtained with $Y_i = 1$. For this reason, our approach implies that BCH codes do not represent a secure choice for McEliece. To our knowledge, it is the first time such a result is proved for the whole class of BCH codes.

We applied this approach to two new cryptosystems [26, 5] that are variants of the McEliece scheme. Both aim at reducing the public keys by using very structured block matrices (cyclic matrices in [5] and dyadic matrices in [26]). We show that our new cryptanalytic point of view is very efficient for all the parameters proposed in [5]. An implementation in MAGMA validates our attack and shows that the private key can be found in a negligible time. For the scheme [26], we are also able to fully recover the private key in almost all cases. An implementation in MAGMA shows that this can be done in time comparable to [5] as long as the dimension m of the solution vector space of the Y_i's is small. In particular, we did not manage to solve efficiently the system when $m = 16$.

Finally, it would be interesting to study if this algebraic approach followed here can be improved in order to mount an attack on the original McEliece cryptosystem. In this case however, there are much more unknowns than in the cases considered here and there is much more freedom left on the Y_i's by looking at the linear equations involving only the Y_i's.

References

1. W. Adams and P. Loustaunau. *An Introduction to Gröbner Bases*. Americal Mathematical Society, 7 1994.
2. R. Avanzi. Lightweight asymmetric cryptography and alternatives to rsa, ecrypt european network of excellence in cryptology ist-2002-507932. `http://www.ecrypt.eu.org/ecrypt1/documents/D.AZTEC.2-1.2.pdf`, 2005.
3. M. Baldi, M. Bodrato, and G.F. Chiaraluce. A new analysis of the McEliece cryptosystem based on QC-LDPC codes. In *Security and Cryptography for Networks (SCN)*, pages 246–262, 2008.
4. M. Baldi and G. F. Chiaraluce. Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In *IEEE International Symposium on Information Theory*, pages 2591–2595, Nice, France, March 2007.
5. T. P. Berger, P.L. Cayrel, P. Gaborit, and A. Otmani. Reducing key length of the McEliece cryptosystem. In Bart Preneel, editor, *Progress in Cryptology - Second International Conference on Cryptology in Africa (AFRICACRYPT 2009)*, volume 5580 of *Lecture Notes in Computer Science*, pages 77–97, Gammarth, Tunisia, June 21-25 2009.
6. T. P. Berger and P. Loidreau. How to mask the structure of codes for a cryptographic use. *Designs Codes and Cryptography*, 35(1):63–79, 2005.
7. T. P. Berger and Pierre Loidreau. Designing an efficient and secure public-key cryptosystem based on reducible rank codes. In *INDOCRYPT*, volume 3348 of *LNCS*, pages 218–229, 2004.
8. D. J. Bernstein, T. Lange, and C. Peters. Attacking and defending the McEliece cryptosystem. In *PQCrypto*, volume 5299 of *LNCS*, pages 31–46, 2008.
9. D. J. Bernstein, T. Lange, C. Peters, and H. van Tilborg. Explicit bounds for generic decoding algorithms for code-based cryptography. In *Pre-proceedings of WCC 2009*, pages 168–180, 2009.
10. B. Biswas and N. Sendrier. McEliece cryptosystem implementation: Theory and practice. In *PQCrypto*, volume 5299 of *LNCS*, pages 47–62, 2008.
11. Buchberger, B. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, Innsbruck, 1965.
12. D. A. Cox, J. B. Little, and D. O'Shea. *Ideals, Varieties, and algorithms: an Introduction to Computational Algebraic Geometry and Commutative Algebra.* Undergraduate Texts in Mathematics, Springer-Verlag, New York., 2001.
13. J.-C. Faugère. A new efficient algorithm for computing gröbner bases (f4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.
14. J.-C. Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero : F5. In *ISSAC'02*, pages 75–83. ACM press, 2002.

15. J.-C. Faugère, F. Levy-dit Vehel, , and L. Perret. Cryptanalysis of minrank. In David Wagner, editor, *Advances in Cryptology - CRYPTO'08*, volume 5157, pages 280–296, 2008.
16. Jean-Charles Faugère, Patrizia M. Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *J. Symb. Comput.*, 16(4):329–344, 1993.
17. C. Faure and L. Minder. Cryptanalysis of the McEliece cryptosystem over hyperelliptic curves. In *Proceedings of the eleventh International Workshop on Algebraic and Combinatorial Coding Theory*, pages 99–107, Pamporovo, Bulgaria, June 2008.
18. M. Finiasz and N. Sendrier. Security bounds for the design of code-based cryptosystems. In M. Matsui, editor, *Asiacrypt 2009*, volume 5912 of *LNCS*, pages 88–105. Springer, 2009.
19. E. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. Ideals over a non-commutative ring and their applications to cryptography. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques*, number 547 in LNCS, pages 482–489, Brighton, april 1991.
20. P. Gaborit. Shorter keys for code based cryptography. In *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*, pages 81–91, Bergen, Norway, March 2005.
21. J. K. Gibson. Severely denting the Gabidulin version of the McEliece public key cryptosystem. *Design Codes and Cryptography*, 6(1):37–45, 1995.
22. H. Janwa and O. Moreno. McEliece public key cryptosystems using algebraic-geometric codes. *Designs Codes and Cryptography*, 8(3):293–307, 1996.
23. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North–Holland, Amsterdam, fifth edition, 1986.
24. R. J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.
25. L. Minder and A. Shokrollahi. Cryptanalysis of the Sidelnikov cryptosystem. In *Eurocrypt 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 347–360, Barcelona, Spain, 2007.
26. R. Misoczki and P. S. L. M. Barreto. Compact McEliece keys from Goppa codes. In *Selected Areas in Cryptography (SAC 2009)*, Calgary, Canada, August 13-14 2009.
27. P. Nguyen. New trends in cryptology, european project "storkstrategic roadmap for advances in cryptology - crypto" ist-2002-38273. `http://www.di.ens.fr/~pnguyen/pub.html#Ng03`, 2003.
28. H. Niederreiter. A public-key cryptosystem based on shift register sequences. In *EUROCRYPT*, volume 219 of *LNCS*, pages 35–39, 1985.
29. A. Otmani, J.P. Tillich, and L. Dallot. Cryptanalysis of McEliece cryptosystem based on quasi-cyclic ldpc codes. In *Proceedings of First International Conference on Symbolic Computation and Cryptography*, pages 69–81, Beijing, China, April 28-30 2008. LMIB Beihang University.
30. R. Overbeck. Structural attacks for public key cryptosystems based on Gabidulin codes. *J. Cryptology*, 21(2):280–301, 2008.
31. V.M. Sidelnikov. A public-key cryptosytem based on Reed-Muller codes. *Discrete Mathematics and Applications*, 4(3):191–207, 1994.
32. V.M. Sidelnikov and S.O. Shestakov. On the insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discrete Mathematics and Applications*, 1(4):439–444, 1992.
33. J. Stern. A method for finding codewords of small weight. In G. D. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1988.
34. C. Wieschebrink. Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes. eprint 452, available at http://eprint.iacr.org/2009/452.pdf, 2009.

A Gröbner Basics.

The classical approach for computing a Gröbner basis of $\mathcal{I} \cap \mathbb{F}_{q^m}[\mathbf{Y}']$ can be described as follows. A reader already familiar with polynomial system solving can skip this part. We have to choose a suitable ordering on the monomials (for a definition of such orders, see for instance [12, Chap. 2, p. 52]). In particular, we have to select an elimination ordering ([1, Chap. 2.3, p. 69]) on the blocks $\mathbf{X}'$, $\mathbf{Y}'$ such that the variables occurring in $\mathbf{X}$ are greater that those of $\mathbf{Y}'$ (denoted by $\mathbf{X}' >> \mathbf{Y}'$). According to [1, Theo. 2.3.4, Chap. 2.3, p. 69], this elimination ordering will permit to compute a Gröbner basis G_{deg} of $\mathcal{I} \cap \mathbb{F}_{q^m}[\mathbf{Y}']$ with respect to a degree order on the variables of $\mathbf{Y}'$ (i.e. this is the order induced when "removing" the variables of the block $\mathbf{X}'$ in the elimination ordering). In theory, to compute the variety $\mathcal{V}'$ associated to $\mathcal{I} \cap \mathbb{F}_{q^m}[\mathbf{Y}']$, we have to perform a change of ordering on G_{deg} to compute Gröbner basis G_{lex} of $\mathcal{I} \cap \mathbb{F}_{q^m}[\mathbf{Y}']$. If we assume that $\mathcal{V}'$ is *zero-dimensional* (i.e. has a finite number of solutions so that $\#\mathcal{V}' < \infty$), then an efficient tool to perform the change of ordering is the FGLM algorithm [16]. The complexity of computing G_{lex} from G_{degree} with FGLM is polynomial in the size of $\mathcal{V}'$, i.e. $\mathcal{O}\big((\#\mathcal{V}')^3\big)$. In our case, the size of $\mathcal{V}'$ is very small (< 10).

We have used a slightly modified version of F_4 [13] for computing a Gröbner basis G_{deg} of $\mathcal{I} \cap \mathbb{F}_{q^m}[\mathbf{Y}']$. The idea is to adapt the algorithm for performing the Gröbner basis computation in $\mathbb{F}_{q^m}[\mathbf{X}'][\mathbf{Y}']$, i.e. the set of polynomials in $\mathbf{Y}'$ whose coefficients are polynomials in $\mathbb{F}_{q^m}[\mathbf{X}']$. As for the usual F_4, we process degree by degree. However, we consider only the degree of the polynomials w.r.t. the variables of $\mathbf{X}'$. We stop the computation as soon as we have sufficiently many equations in $\mathbf{Y}'$ (for instance, as soon as we detect that $\mathcal{V}'$ has a finite number of solution, i.e. of dimension zero). The modified version is defined below.

1 Algorithm F_4 (modified version)

Input: $\begin{cases} \mathbf{X}\text{' and } \mathbf{Y}\text{'} \\ F \text{ a finite subset of } \mathbb{F}_{q^m}[\mathbf{X}', \mathbf{Y}'] \\ < \text{ a monomial admissible order} \end{cases}$
Output: a finite subset of $\mathbb{F}_{q^m}[\mathbf{Y}']$.
$G := F$ and $P := \{\text{CritPair}(f, g) \mid (f, g) \in G^2 \text{ with } f \neq g\}$
while $P \neq \emptyset$ and $\mathsf{dim}(G \cap \mathbb{F}_{q^m}[\mathbf{Y}']) > 0$ **do**
  $d := \min\{\deg_{\mathbf{X}'}(p) \mid p \in P\}$ minimal partial degree of critical pairs
  Extract from P, P_d the list of critical pairs of degree d
  $R :=$MATRIX_REDUCTION$(\text{Left}(P_d) \cup \text{Right}(P_d), G)$
  for $h \in R$ **do**
    $P := P \cup \{\text{CritPair}(h, g) \mid g \in G\}$
    $G := G \cup \{h\}$
return $G \cap \mathbb{F}_{q^m}[\mathbf{Y}']$

For the definition of MATRIX_REDUCTION, and CritPair, we refer to [13]. Briefly, the first function performs the usual polynomial reduction of Buchberger's algorithm [12] using linear algebra. The second function selects critical pairs with respect to a defined strategy.

B Proof of Proposition 5

Lemma 1. *Let $N = N_0\ell$ with $\ell = 2^e$ for some integer e and let $\boldsymbol{h}$ be a vector in $\mathbb{F}_{q^m}^N$ that satisfies Equation (17). Let $\mathscr{G}(\boldsymbol{a}^*, G)$ be the Goppa code such that $\boldsymbol{a}^* = (a_0^*, \ldots, a_{N-1}^*)$ is defined by $a_j^* = 1/h_j + 1/h_0$ for $0 \leq j \leq N-1$ and $G(X) = \prod_{i=0}^{\ell-1}(X - z_i)$ with $z_i = 1/h_i$. Then for any i, j in $\{0, \ldots, N-1\}$ we have $a_{i\oplus j}^* = a_i^* + a_j^*$ and for any $0 \leq j \leq N_0 - 1$ and $0 \leq i, i' \leq \ell - 1$*

$$G(a_{j\ell+i}^*)^{-1} = \prod_{l=j\ell}^{(j+1)\ell-1} h_l$$

Proof. The property that $a_{i\oplus j}^* = a_i^* + a_j^*$ comes from Equation (17). Furthermore, we have:

$$G(a_{j\ell+i}^*)^{-1} = \prod_{\ell=0}^{\ell-1}(z_\ell - a_{j\ell+i}^*)^{-1} = \prod_{\ell=0}^{\ell-1}(1/h_\ell + 1/h_{j\ell+i} + 1/h_0)^{-1} = \prod_{\ell=0}^{\ell-1} h_{j\ell+\ell}$$

which terminates the proof.

We remark in particular that we have $G(a_{j\ell+i}^*) = G(a_{j\ell}^*)$ for any $0 \leq j \leq n_0 - 1$ and $0 \leq i \leq \ell - 1$. The next lemma we give without proof shows that the action of a dyadic permutation can be simply characterized as a translation.

Lemma 2. *Let t and d two integers such that $0 \leq d \leq \ell - 1$. For any vector $\boldsymbol{v} = (v_0, \ldots, v_{\ell-1})$, we have:*

$$\boldsymbol{v} \times \boldsymbol{\Delta}_\ell(\boldsymbol{b}_d) = (v_d, v_{1\oplus d}, \ldots, v_{(\ell-1)\oplus d}) \tag{21}$$

where the vector $\boldsymbol{b}_d = (b_{d,0}, \ldots, b_{d,\ell-1})$ is such that $b_{d,j} = 0$ if $j \neq d$ and $b_{d,d} = 1$.

We are now prepared to prove Proposition 5. Let $(\boldsymbol{h}, \boldsymbol{\omega}, \boldsymbol{d}, \boldsymbol{\gamma})$ be the private key and let $\boldsymbol{G}$ be the public generator matrix. We shall see that a parity-check matrix for the code generated by $\boldsymbol{G}$ is $\boldsymbol{V}_\ell(\boldsymbol{a}, \boldsymbol{\lambda})$ with

$$\begin{aligned} a_{j\ell+i} &= a^*_{(\omega_j t+i)\oplus d_\ell} \\ \lambda_{j\ell+i} &= \gamma_j \, G(a^*_{\omega_j t})^{-1} \end{aligned}$$

where $\boldsymbol{a}^*$ and $G(X)$ are defined as in Lemma 1. Indeed, we know that the code defined by the parity-check matrix $\boldsymbol{\Delta}_\ell(\boldsymbol{h})$ is also defined by the parity-check matrix $\boldsymbol{V}_\ell(\boldsymbol{a}, \boldsymbol{\lambda})$ where $\lambda_j = G(a_j)^{-1}$ for any $0 \leq j \leq N-1$. Recall from Lemma 1 that $G(a_{j\ell+i}) = G(a_{j\ell})$ for any $0 \leq j \leq N_0 - 1$ and $0 \leq i \leq \ell - 1$. The role of $\boldsymbol{\omega}$ is to pick n_0 dyadic blocks from $\boldsymbol{\Delta}_\ell(\boldsymbol{h})$. These blocks correspond to the columns $a^*_{\omega_j\ell}, \ldots, a^*_{(\omega_j+1)\ell-1}$ of $\boldsymbol{V}_\ell(\boldsymbol{a}, \boldsymbol{\lambda})$ when j describes $\{1, \ldots, n_0\}$. These columns are then multiplied by a dyadic permutation matrix $\boldsymbol{\Delta}_\ell(\boldsymbol{b}_{d_\ell})$ which leads to reorder the columns as $a_{\omega_j\ell\oplus d_j}, \ldots, a_{((\omega_j+1)\ell-1)\oplus d_j}$ according to Lemma 2. Finally, each dyadic block is scaled by γ_j which means that if we set $\lambda_{j\ell+i} = \gamma_j \, G(a_{\omega_j\ell})^{-1}$ then $\boldsymbol{V}_\ell(\boldsymbol{a}, \boldsymbol{\lambda})$ is another parity-check matrix of the code generated by $\boldsymbol{G}$.

We are now going to show that for any $0 \leq j \leq n_0 - 1$ and $0 \leq i, i' \leq \ell - 1$, we have the following equations:

$$\begin{cases} \lambda_{j\ell+i} & = \lambda_{j\ell} \\ a_{j\ell+i} + a_{j\ell} & = a_i + a_0 \\ a_{j\ell+i\oplus i'} & = a_{j\ell+i} + a_{j\ell+i'} + a_{j\ell} \end{cases} \tag{22}$$

It is clear from Lemma 1 that $\lambda_{j\ell+i} = \lambda_{j\ell}$. On the other hand, $a_{j\ell+i} = a_{(\omega_j\ell+i)\oplus d_j} = 1/h_{(\omega_j\ell+i)\oplus d_j} + 1/h_0$. From Equation (17) we thus have:

$$\begin{aligned} a_{j\ell+i} &= \frac{1}{h_{\omega_j\ell+i}} + \frac{1}{h_{d_j}} \\ &= \frac{1}{h_{\omega_j\ell}} + \frac{1}{h_i} + \frac{1}{h_0} + \frac{1}{h_{d_j}} \\ &= \frac{1}{h_{\omega_j\ell\oplus d_j}} + \frac{1}{h_i} + \frac{1}{h_0} \\ &= a_{j\ell} + \frac{1}{h_i} + \frac{1}{h_0}. \end{aligned}$$

We observe in particular that $a_i + a_0 = 1/h_i + 1/h_0$ and since this quantity does not depend on ℓ, this is equivalent to say that $a_{j\ell+i} + a_{j\ell} = a_i + a_0$. Before proving the third equation, we can first see that $a_{j\ell+i\oplus i'} + a_{j\ell} = a_{i\oplus i'} + a_0$. So if we know that $a_{i\oplus i'} = a_i + a_{i'} + a_0$ then we would get $a_{i\oplus i'} = a_{j\ell+i} + a_{j\ell} + a_{i'}$ which finally implies $a_{i\oplus i'} = a_{j\ell+i} + a_{j\ell+i'} + a_0$ that leads to the expected result. Now we have $a_{i\oplus i'} = a_{(\omega_1\ell+i+i')\oplus d_1} = a_{\omega_1\ell+i+i'} + a_{d_1} = a_{\omega_1\ell+i} + a_{i'} + a_{d_1} = a_{(\omega_1\ell+i)\oplus d_1} + a_{i'}$. Therefore we obtain:

$$\begin{aligned} a_{i\oplus i'} &= a_i + a_{i'} + a_{\omega_1\ell} + a_{d_1} + a_{\omega_1\ell} + a_{d_1} \\ &= a_i + a_{i'} + a_0. \end{aligned}$$