An efficient structural attack on NIST submission DAGS

Élise Barelli*[1] and Alain Couvreur†[1]

[1]INRIA & LIX, CNRS UMR 7161
École polytechnique, 91128 Palaiseau Cedex, France.

Abstract

We present an efficient key recovery attack on code based encryption schemes using some quasi–dyadic alternant codes with extension degree 2. This attack permits to break the proposal DAGS recently submitted to NIST.

keywords : Code-based Cryptography, McEliece encryption scheme, Key recovery attack, Alternant codes, Quasi–dyadic codes, Schur product of codes.

Introduction

In 1978, in the seminal article [23], R. J. McEliece designed a public key encryption scheme relying on the hardness of the bounded decoding problem [7], *i.e.* on the hardness of decoding an arbitrary code. For a long time, this scheme was considered as unpractical because of the huge size of the public keys compared to public key encryption schemes relying on algorithmic number theoretic problems. Hence, for a long time, code based cryptography was considered as a purely theoretic area with few perspectives of practical applications. The trend changed in the last decade because of the progress of quantum computing and the increasing threat of the existence in a near future of a quantum computer able to break usual cryptography primitives based on number theoretic problems : integer factorisation and discrete logarithm in finite fields or elliptic curves. An evidence for this change of trend is the recent call of the National Institute for Standards and Technology (NIST). for post quantum cryptography. The majority of the submissions to this call are based either on codes or on lattices.

After forty years of research on code based cryptography, one can identify two general trends for instantiating McEliece's scheme. The first one consists in using codes from probabilistic constructions such as LDPC and MDPC codes [25, 1]. The other one consists in using algebraic codes such as Goppa codes or more generally alternant codes.

Concerning McEliece instantiations based on algebraic codes, which include McEliece's original proposal based on binary Goppa codes, two approaches have been considered in order to address the drawback of the large of pubic key sizes. On the one hand, some proposals suggested to replace Goppa or alternant codes by more structured codes such as generalised Reed–Solomon (GRS) codes [26], their low dimensional subcodes [6], or GRS codes to which

*elise.barelli@inria.fr
†alain.couvreur@lix.polytechnique.fr

various transformations have been applied [31, 2, 30]. It turns out that most of these proposals have been subject to polynomial time key-recovery attacks [29, 32, 10, 14]. In addition, proposals based on Goppa codes which are *close* to GRS codes, namely Goppa code with a low extension degree m have been the target of some structural attacks [19, 13]. On the other hand, many proposals suggest the use of codes with a non trivial automorphism group [20, 5, 24, 28]. A part of these proposals have been either partially or completely broken [27, 18, 17]. In particular, in the design of such proposal, precautions should be taken since the knowledge of a non trivial automorphism group of the public code facilitates algebraic attacks by significantly reducing the degrees and number of variables of the algebraic system to solve to in order to recover the secret key.

Among the recent submissions to NIST call for post quantum cryptography, a proposal called DAGS [3] is based on the use of quasi–dyadic (QD) generalised Srivastava codes with extension degree $m = 2$. By *quasi–dyadic* we mean that the permutation group of the code is of the form $(\mathbb{Z}/2\mathbb{Z})^\gamma$ for some positive integer γ. Moreover, generalised Srivastava codes form a proper subclass of alternant codes. DAGS proposal takes advantage of both usual techniques to reduce the size of the keys. First, by using alternant codes which are close to generalised Reed Solomon codes *i.e.* with an extension degree 2. Second, by using codes with a large permutation group. In terms of security with respect to key recovery attacks, DAGS parameters are chosen to be out of reach of the algebraic attacks [18, 17]. In addition, it should be emphasised that the choice of alternant codes which are not Goppa codes permits to be out of reach of the distinguisher by shortening and squaring used in [13].

Our contribution In this article, we present an attack breaking McEliece instantiations based on alternant codes with extension degree 2 and a large permutation group. This attack permits to recover the secret key in $O\left(n^{3+\frac{2q}{|\mathcal{G}|}}\right)$ operations in $\mathbb{F}_q$, where $\mathcal{G}$ denotes the permutation group and n the code length. The key step of the attack consists in computing some specific code referred to as the *norm trace code*, from which the secret key can easily be recovered. For this main step, we present two ways to proceed, the first approach is based on a partial brute force search while the second one is based on the resolution of a bilinear system. An analysis of the work factor of this attack using the first approach shows that DAGS keys with respective estimated security levels 128, 192 and 256 bits can be broken with respective approximate work factors $2^{70}, 2^{80}$ and 2^{58}. For the second approach, we were not able to provide a complexity analysis. However, its practical implementation using Magma [8] is impressively efficient on some DAGS parameters. In particular, it permits to break claimed 256 bits security keys in less than one minute!

This attack is a novel and original manner to recover the structure of alternant codes by jointly taking advantage the permutation group and the small size of the extension degree. Even if some variant of the attack reposes on the resolution of a bilinear system, this system has nothing to do with those of algebraic attacks of [18, 17, 19]. On the other hand, despite this attack shares some common points with that of [13] where the computation of the norm trace code is also an intermediary step, the way we obtain this norm trace code and the reasons why it is possible to compute it are completely different. In particular, the keys we break in the present article are out of reach of a distingusher by shortening and squaring and hence our attack differs from filtration attacks as in [13, 11].

Outline of the article This article is organised as follows. Prerequisites on algebraic codes and quasi–dyadic codes are given in Section 1. In Section 2, we recall the definition of the Schur product of codes and some of its properties. In Section 3, we introduce a fundamental object for this attack called the *conductor* of a pair of codes. The description of the attack is given in Section 4 and its complexity is discussed in Section 5. Finally, Section 6 is devoted to the implementation of the attack and the presentation of experimental results.

1 Notation and prerequisites

1.1 Subfield subcodes and trace codes

Definition 1. Given a code $\mathscr{C}$ of length n over $\mathbb{F}_{q^m}$, its *subfield subcode* is the subcode of vectors whose entries all lie in $\mathbb{F}_q$, that is the code:

$$\mathscr{C} \cap \mathbb{F}_q^n.$$

The *trace code* is the image of the code by the component wise trace map

$$\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\mathscr{C}) \stackrel{\text{def}}{=} \left\{ \mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\boldsymbol{c}) \mid \boldsymbol{c} \in \mathscr{C} \right\}.$$

Let us remind a classical and well–known result on subfield subcodes and trace codes.

Theorem 2 (Delsarte Theorem [16])**.** *Let $\mathscr{C} \subseteq \mathbb{F}_{q^m}^n$ be a code. Then*

$$(\mathscr{C} \cap \mathbb{F}_q^n)^\perp = Tr_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\mathscr{C}^\perp).$$

1.2 Generalised Reed–Solomon codes and alternant codes

Notation 1. Let q be a power of prime and k a positive integer. We denote by $\mathbb{F}_q[z]_{<k}$ the vector space of polynomials over $\mathbb{F}_q$ whose degree is bounded from above by k. Let m be a positive integer, we will consider codes over $\mathbb{F}_{q^m}$ with their subfield subcodes over $\mathbb{F}_q$. In § 2 and further, we will focus particularly on the case $m = 2$.

Definition 3 (Supports and multipliers)**.** A vector $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ whose entries are pairwise distinct is called a *support*. A vector $\boldsymbol{y} \in \mathbb{F}_{q^m}^n$ whose entries are all nonzero is referred to as a *multiplier*.

Definition 4 (Generalised Reed–Solomon codes)**.** Let n be a positive integer, $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ be a support and $\boldsymbol{y} \in \mathbb{F}_{q^m}^n$ be a multiplier. The *generalised Reed–Solomon (GRS) code with support $\boldsymbol{x}$ and multiplier $\boldsymbol{y}$ of dimension k* is defined as

$$\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \{(y_1 f(x_1), \ldots, y_n f(x_n)) \mid f \in \mathbb{F}_q[z]_{<k}\}.$$

When $\boldsymbol{y} = \mathbf{1}$, then the code is a *Reed–Solomon code* and is denoted as $\mathbf{RS}_k(\boldsymbol{x})$.

Definition 5 (Alternant code)**.** Let $m,\ n$ be positive integers such that $n \leqslant q^m$. Let $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ be a support, $\boldsymbol{y} \in \mathbb{F}_{q^m}^n$ be a multiplier and r be a positive integer. The *alternant code of support $\boldsymbol{x}$, multiplier $\boldsymbol{y}$ and degree r over $\mathbb{F}_q$* is defined as

$$\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^\perp \cap \mathbb{F}_q^n.$$

The integer m is referred to as the *extension degree of* the alternant code.

Proposition 6. *In the context of Definition 5, we have*

$$\dim \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) \geqslant n - mr.$$

Proof. See [22, Chapter 12]. □

Note that the dual of a GRS code is a GRS code too. This is explicited in Lemma 7 below. Let us first introduce an additional notation.

Notation 2. Let $\boldsymbol{x} \subseteq \mathbb{F}_{q^m}^n$ be a vector with distinct entries, we define the *locator polynomial* $\pi_{\boldsymbol{x}} \in \mathbb{F}_{q^m}[z]$ as

$$\pi_{\boldsymbol{x}}(z) \stackrel{\text{def}}{=} \prod_{i=0}^{n-1}(z - x_i).$$

Lemma 7. *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_{q^m}^n$ be a support and a multiplier of length n and $k \leqslant n$. Then*

$$\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})^\perp = \mathbf{GRS}_{n-k}(\boldsymbol{x}, \boldsymbol{y}^\perp),$$

where

$$\boldsymbol{y}^\perp \stackrel{def}{=} \left(\frac{1}{\pi'_{\boldsymbol{x}}(x_1)y_1}, \ldots, \frac{1}{\pi'_{\boldsymbol{x}}(x_n)y_n} \right),$$

and $\pi'_{\boldsymbol{x}}$ denotes the derivative of the polynomial $\pi_{\boldsymbol{x}}$.

As a direct consequence of Lemma 7 and Definition 5, we get the following explicit description of an alternant code.

$$\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) = \left\{ \left(\frac{1}{\pi'_{\boldsymbol{x}}(x_i)y_i} f(x_i) \right)_{i=1,\ldots,n} \;\middle|\; f \in \mathbb{F}_{q^m}[z]_{<n-r} \right\} \cap \mathbb{F}_q^n. \tag{1}$$

Next, by duality and using Delsarte's Theorem (Theorem 2), we have

$$\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp = \mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}\left(\left\{ (y_i g(x_i))_{i=1,\ldots,n} \;\middle|\; g \in \mathbb{F}_{q^m}[z]_{<r} \right\} \right), \tag{2}$$

where $\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}$ denotes the component wise trace map.

1.2.1 Decoding alternant codes

Alternant codes come with an efficient decoding algorithm. For instance, see [22, Chapter 12§9].

Fact 1. *Given an alternant code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$, there exists an efficient decoding algorithm correcting up to $t = \lfloor \frac{r}{2} \rfloor$ errors. This decoding algorithm can be built from the knowledge of the pair $(\boldsymbol{x}, \boldsymbol{y})$.*

1.2.2 Fully non degenerated alternant codes

We conclude this subsection on alternant codes by a definition which is useful in the sequel.

Definition 8. An alternant code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ is said to be *fully non degenerated* if it satisfies the two following conditions.

(i) A generator matrix of $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ has no zero column.

(ii) $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) \neq \mathscr{A}_{r+1}(\boldsymbol{x}, \boldsymbol{y})$.

It should be emphasised that, in general, an alternant code is fully non degenerated.

1.3 Punctured and shortened codes

The notions of *puncturing* and *shortening* are classical ways to build new codes from existing ones. We recall here their definition.

Definition 9. Let $\mathscr{C}$ be a code of length n and $\mathcal{I} \subseteq \{1, \dots, n\}$. The *puncturing of* $\mathscr{C}$ *at* $\mathcal{I}$ is defined as the code

$$\mathcal{P}_{\mathcal{I}}(\mathscr{C}) \stackrel{\text{def}}{=} \{(c_i)_{i \in \{1,\dots,n\} \setminus \mathcal{I}} \mid \boldsymbol{c} \in \mathscr{C}\}.$$

Definition 10. Let $\mathscr{C}$ be a code of length n and $\mathcal{I} \subseteq \{1, \dots, n\}$. The *shortening of* $\mathscr{C}$ *at* $\mathcal{I}$ is defined as the code

$$\mathcal{S}_{\mathcal{I}}(\mathscr{C}) \stackrel{\text{def}}{=} \mathcal{P}_{\mathcal{I}}\left(\{\boldsymbol{c} \in \mathscr{C} \mid \forall i \in \mathcal{I},\ c_i = 0\}\right).$$

Let us finish by reminding the following classical result.

Notation 3. Let $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ be a vector and $\mathcal{I} \subseteq \{1, \dots, n\}$. Then, the vector $\boldsymbol{x}_{\mathcal{I}}$ denotes the vector obtained from $\boldsymbol{x}$ be removing the entries whose indexes are in $\mathcal{I}$.

Proposition 11. *Let m, r be positive integers. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_{q^m}^n$ be as in Definition 5. Let $\mathcal{I} \subseteq \{1, \dots, n\}$. Then*

$$\mathcal{S}_{\mathcal{I}}(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})) = \mathscr{A}_r(\boldsymbol{x}_{\mathcal{I}}, \boldsymbol{y}_{\mathcal{I}}).$$

Proof. See for instance [13, Proposition 9]. □

1.4 Quasi–dyadic codes, quasi-dyadic alternant codes

Quasi–dyadic (QD) codes are codes with a nontrivial permutation group which is isomorphic to $(\mathbb{Z}/2\mathbb{Z})^\gamma$ for some positive integer γ. Such a code has length $n = n_0 s$ where $s = 2^\gamma$. The permutation group of the code is composed of permutations which are products of transpositions with disjoint supports. The example of interest in the present article is the case of QD–alternant codes. In what follows, we explain how to create such QD–alternant codes.

Notation 4. From now on, q denotes a power of 2 and ℓ denotes the positive integer such that $q = 2^\ell$.

- Let $\mathcal{G} \subset \mathbb{F}_{q^m}$ be an additive subgroup with γ generators, i.e. $\mathcal{G}$ is an $\mathbb{F}_2$–vector subspace of $\mathbb{F}_{q^m}$ of dimension γ with an $\mathbb{F}_2$–basis $a_1, \dots, a_\gamma$. Clearly, as an additive group, $\mathcal{G}$ is isomorphic to $(\mathbb{Z}/2\mathbb{Z})^\gamma$. The group $\mathcal{G}$ acts on $\mathbb{F}_{q^m}$ by translation: for any $a \in \mathcal{G}$, we denote by τ_a the translation

$$\tau_a : \left\{ \begin{array}{ccc} \mathbb{F}_{q^m} & \longrightarrow & \mathbb{F}_{q^m} \\ x & \longmapsto & x + a \end{array} \right. .$$

- Using the basis $(a_1, \dots, a_\gamma)$, we fix an ordering in $\mathcal{G}$ as follows. Any element of $\mathcal{G}$ is a linear combination $u_1 a_1 + \cdots + u_\gamma a_\gamma$ where the u_i's are elements of $\mathbb{Z}/2\mathbb{Z}$. Thus, any element $u_1 a_1 + \cdots + u_\gamma a_\gamma \in \mathcal{G}$ can be regarded as an element $(u_1, \dots, u_\gamma) \in (\mathbb{Z}/2\mathbb{Z})^\gamma$ and we sort them by lexicographic order. For instance, if $\gamma = 3$:

$$0 < a_1 < a_2 < a_1 + a_2 < a_3 < a_1 + a_3 < a_2 + a_3 < a_1 + a_2 + a_3.$$

- Let $n = 2^\gamma n_0$ for some positive n_0 and such that $n \leqslant q^m$. Let $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ be a support which splits into n_0 blocks of 2^γ elements of $\mathbb{F}_{q^m}$, each block being an orbit under the action of $\mathcal{G}$ by translation on $\mathbb{F}_{q^m}$ sorted using the previously described ordering. For instance, suppose $\gamma = 2$, then such an $\boldsymbol{x}$ is of the form,

$$\begin{aligned} \boldsymbol{x} = \quad & (t_1, t_1 + a_1, t_1 + a_2, t_1 + a_1 + a_2, \ldots, \\ & \ldots, t_{n_0}, t_{n_0} + a_1, t_{n_0} + a_2, t_{n_0} + a_1 + a_2), \end{aligned} \tag{3}$$

  where the t_i's are chosen to have disjoint orbits under the action of $\mathcal{G}$ by translation on $\mathbb{F}_{q^m}$.

- Let $\boldsymbol{y} \in \mathbb{F}_{q^m}^n$ be a multiplier which also splits into n_0 blocks of length 2^γ whose entries are equal.

- Let r be a positive integer and consider the code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$.

- The set of entries of $\boldsymbol{x}$ is globally invariant under the action of $\mathcal{G}$ by translation. In particular, for any $a \in \mathcal{G}$, the translation τ_a induces a permutation of the code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$. This permutation is a product of $\frac{n}{2}$ transpositions with disjoint supports. We refer this permutation to as σ_a. For instance, reconsidering the example (3), the permutations σ_{a_1} and $\sigma_{a_1+a_2}$ are respectively of the form

$$\begin{aligned} \sigma_{a_1} &= (1,2)(3,4)\cdots(n-3,n-2)(n-1,n) \\ \sigma_{a_1+a_2} &= (1,4)(2,3)\cdots(n-3,n)(n-2,n-1). \end{aligned}$$

  The group of permutations $\{\sigma_a \mid a \in \mathcal{G}\}$ is isomorphic to $\mathcal{G}$ and hence to $(\mathbb{Z}/2\mathbb{Z})^\gamma$. Since it is isomorphic to $\mathcal{G}$ and in order to limit the amount of notation, we also denote this group of permutations by $\mathcal{G}$.

Proposition 12. *For any positive integer r, the code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ is quasi–dyadic.*

Proof. See for instance [15, Chapter 5]. □

1.5 Invariant subcode of a quasi–dyadic code

Definition 13. Given a code $\mathscr{C}$ with a non–trivial permutation group $\mathcal{G}$, we define the code $\mathscr{C}^{\mathcal{G}}$ as the subcode of $\mathscr{C}$:

$$\mathscr{C}^{\mathcal{G}} \stackrel{\text{def}}{=} \{\boldsymbol{c} \in \mathscr{C} \mid \forall \sigma \in \mathcal{G},\ \sigma(\boldsymbol{c}) = \boldsymbol{c}\}.$$

The invariant subcode has repeated entries since on any orbit of the support under the action of $\mathcal{G}$, the entries of a codeword are equal. This motivates an alternative definition of the invariant code where repetitions have been removed.

Definition 14. In the context of Definition 13, let $\boldsymbol{c} \in \mathbb{F}_{q^m}^n$ be a vector such that for any $\sigma \in \mathcal{G}$, $\sigma(\boldsymbol{c}) = \boldsymbol{c}$. We denote by $\overline{\boldsymbol{c}}$ the vector obtained by keeping only one entry per orbit under the action of $\mathcal{G}$ on the support. Next we define the *invariant code with non repeated entries* as

$$\overline{\mathscr{C}^{\mathcal{G}}} \stackrel{\text{def}}{=} \left\{\overline{\boldsymbol{c}} \mid \boldsymbol{c} \in \mathscr{C}^{\mathcal{G}}\right\}.$$

We are interested in the structure of invariant of QD alternant codes. To study this structure, we first need to recall some basic notions of additive polynomials.

1.5.1 Additive polynomials

Definition 15. An *additive polynomial* $P \in \mathbb{F}_{q^m}[z]$ is a polynomial whose monomials are all of the form z^{2^i} for $i \geqslant 0$. Such a polynomial satisfies $P(a+b) = P(a)+P(b)$ for any $a, b \in \mathbb{F}_{q^m}$.

The zero locus of an additive polynomial in $\mathbb{F}_{q^m}$ is an additive subgroup of $\mathbb{F}_{q^m}$ and such polynomials satisfy some interpolation properties.

Proposition 16. *Let $\mathcal{G} \subset \mathbb{F}_{q^m}$ be an additive group of cardinality 2^γ. There exists a unique additive polynomial $\psi_{\mathcal{G}} \in \mathbb{F}_{q^m}[z]$ which is monic of degree 2^γ and vanishes at any element of $\mathcal{G}$.*

Proof. Let $a_1, \ldots, a_\gamma$ be a set of generators of $\mathcal{G}$. The polynomial $\psi_{\mathcal{G}}$ can be constructed using the so–called *Moore determinant*:

$$\psi_{\mathcal{G}}(z) \stackrel{\text{def}}{=} \begin{vmatrix} a_1 & \cdots & a_\gamma \\ \vdots & & \vdots \\ a_1^{2^{\gamma-1}} & \cdots & a_\gamma^{2^{\gamma-1}} \end{vmatrix}^{-1} \cdot \begin{vmatrix} a_1 & \cdots & a_\gamma & z \\ a_1^2 & \cdots & a_\gamma^2 & z^2 \\ \vdots & & \vdots & \vdots \\ a_1^{2^\gamma} & \cdots & a_\gamma^{2^\gamma} & z^{2^\gamma} \end{vmatrix}.$$

See [21, Proposition 1.3.5] for further details. □

Notation 5. From now on, given an additive subgroup $\mathcal{G} \subseteq \mathbb{F}_{q^m}$, we always denote by $\psi_{\mathcal{G}}$ the unique monic additive polynomial of degree $|\mathcal{G}|$ in $\mathbb{F}_{q^m}[z]$ which vanishes on $\mathcal{G}$.

1.5.2 Invariant of a quasi–dyadic alternant code

It turns out the invariant code of a QD alternant code (after puncturing repeated entries) is an alternant code too. This relies on the following classical result of invariant theory for which a simple proof can be found in [17].

Theorem 17. *Let $f \in \mathbb{F}_{q^m}[z]$ and $\mathcal{G} \subset \mathbb{F}_{q^m}$ be an additive subgroup. Suppose that for any $a \in \mathcal{G}$, $f(z) = f(z+a)$. Then, there exists $h \in \mathbb{F}_{q^m}[z]$ such that $f(z) = h(\psi_{\mathcal{G}}(z))$, where $\psi_{\mathcal{G}}$ is the monic additive polynomial of degree $|\mathcal{G}|$ vanishing at any element of $\mathcal{G}$.*

This entails the following result on the structure of the invariant code of an alternant code. We refer to Definition 14 for the notation in the following statement.

Theorem 18. *Let $\mathscr{C} = \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ be a QD–alternant code with permutation group $\mathcal{G}$ of order 2^γ. Set $r' = \left\lfloor \frac{r}{2^\gamma} \right\rfloor$. Then,*

$$\overline{\mathscr{C}}^{\mathcal{G}} = \mathscr{A}_{r'}(\overline{\psi_{\mathcal{G}}(\boldsymbol{x})}, \overline{\boldsymbol{y}}),$$

Proof. See [17] or [4]. □

1.6 McEliece encryption scheme

Here we remind how McEliece public key encryption scheme instantiated with alternant codes works.

Public key A pair $(\boldsymbol{G}, t)$ where $\boldsymbol{G} \in \mathcal{M}_{k\times n}(\mathbb{F}_q)$ is a generator matrix of an alternant code $\mathscr{C}_{\text{pub}} = \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ and $t = \lfloor \frac{r}{2} \rfloor$ is the number of errors one can correct.

Secret key The pair $(\boldsymbol{x}, \boldsymbol{y})$ whose knowledge permits to decode.

Encryption A plain text $\boldsymbol{m} \in \mathbb{F}_q^k$ is encrypted as $\boldsymbol{c} = \boldsymbol{mG} + \boldsymbol{e}$ where $\boldsymbol{e}$ is a uniformly random vector of $\mathbb{F}_q^n$ of weight $\leqslant t$.

Decryption Using the decoding algorithm, compute $\boldsymbol{mG}$ from $\boldsymbol{c}$. Then deduce $\boldsymbol{m}$ by Gaussian elimination.

A key–recovery attack consists in recovering a pair $(\boldsymbol{x}', \boldsymbol{y}')$ such that $\mathscr{C}_{\text{pub}} = \mathscr{A}_r(\boldsymbol{x}', \boldsymbol{y}')$.

1.7 DAGS

Among the schemes recently submitted to NIST, the submission DAGS [3] uses as a primitive a McEliece encryption scheme based on QD generalised Srivastava codes. It is well known that generalised Srivastava codes form a subclass of alternant codes [22, Chapter 12]. Therefore, this proposal lies in the scope of the attack presented in what follows.

Parameters proposed in DAGS submission are listed in Table 1.

Name	q	m	n	n_0	k	k_0	γ	r_0
`DAGS_1`	2^5	2	832	52	416	26	4	13
`DAGS_3`	2^6	2	1216	38	512	16	5	11
`DAGS_5`	2^6	2	2112	33	704	11	6	11

Table 1: Parameters proposed in DAGS.

Let us remind what parameters $q, m, n, n_0, k, k_0, \gamma, r_0$ stand for:

- q denotes the size of the base field of the alternant code;
- m denotes the extension degree. Hence the GRS code above the alternant code is defined over $\mathbb{F}_{q^m}$;
- n denotes the length of the QD alternant code;
- n_0 denotes the length of the invariant subcode, i.e. $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\mathcal{G}}$, where $\mathcal{G}$ denotes the permutation group.
- k denotes the dimension of the QD alternant code;
- k_0 denotes the dimension of the invariant code;
- γ denotes the number of generators of $\mathcal{G}$, *i.e.* $\mathcal{G} \simeq (\mathbb{Z}/2\mathbb{Z})^\gamma$;
- r_0 denotes the degree of the invariant code, which is alternant according to Theorem 18.

Remark 1. The indexes `1`, `3` and `5` in the parameters names correspond to security levels according to NIST's call. Level 1, corresponds to 128 bits security with a classical computer, Level 3 to 192 bits security and Level 5 to 256 bits security.

In addition to the set of parameters of Table 1, we introduce self chosen smaller parameters listed in Table 2. They **do not** correspond to claimed secure instantiations of the scheme but permitted to test some of our assumptions by computer aided calculations.

Name	q	m	n	n_0	k	k_0	γ	r_0
DAGS_0	2^4	2	240	15	80	5	4	5

Table 2: Small scale parameters, **not** proposed in DAGS.

2 Schur products

From now on and unless otherwise specified, the extension degree m will be equal to 2. This is the context of any proposed parameters in DAGS.

2.1 Product of vectors

The component wise product of two vectors in $\mathbb{F}_q^n$ is denoted by

$$\boldsymbol{a} \star \boldsymbol{b} \stackrel{\text{def}}{=} (a_1 b_1, \ldots, a_n b_n).$$

Next, for any positive integer t we define $\boldsymbol{a}^{\star t}$ as

$$\boldsymbol{a}^{\star t} \stackrel{\text{def}}{=} \underbrace{\boldsymbol{a} \star \cdots \star \boldsymbol{a}}_{t \text{ times}}.$$

More generally, given a polynomial $P \in \mathbb{F}_q[z]$ we defined $P(\boldsymbol{a})$ as the vector $(P(a_1), \ldots, P(a_n))$. In particular, given $\boldsymbol{a} \in \mathbb{F}_{q^2}^n$, we denote by $\mathrm{Tr}(\boldsymbol{a})$ and $\mathrm{N}(\boldsymbol{a})$ the vectors obtained by applying respectively the trace and the norm map component by component:

$$\mathrm{Tr}(\boldsymbol{a}) \stackrel{\text{def}}{=} (a_1 + a_1^q, \ldots, a_n + a_n^q)$$
$$\mathrm{N}(\boldsymbol{a}) \stackrel{\text{def}}{=} (a_1^{q+1}, \ldots, a_n^{q+1}).$$

Finally, the all one vector $(1, \ldots, 1)$, which is the unit vector of the algebra $\mathbb{F}_q^n$ with operations $+$ and $\star$ is denoted by $\mathbf{1}$.

2.2 Schur product of codes

The *Schur product* of two codes $\mathscr{A}$ and $\mathscr{B} \subseteq \mathbb{F}_q^n$ is defined as

$$\mathscr{A} \star \mathscr{B} \stackrel{\text{def}}{=} \langle \boldsymbol{a} \star \boldsymbol{b} \mid \boldsymbol{a} \in \mathscr{A},\ \boldsymbol{b} \in \mathscr{B} \rangle_{\mathbb{F}_q}.$$

In particular, $\mathscr{A}^{\star 2}$ denotes the *square code* of a code $\mathscr{A}$: $\mathscr{A}^{\star 2} \stackrel{\text{def}}{=} \mathscr{A} \star \mathscr{A}$.

2.3 Schur products of GRS and alternant codes

The behaviour of GRS and of some alternant codes with respect to the Schur product is very different from that of random codes. This provides a manner to distinguish GRS codes from random ones and leads to a cryptanalysis of GRS based encryption schemes [32, 10, 14]. Some alternant codes, namely Wild Goppa codes with extension degree 2 have been also subject to a cryptanalysis based on Schur products computations [12, 13].

Here we remind an elementary but crucial result.

Theorem 19. *Let $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ be a support and $\boldsymbol{y}, \boldsymbol{y}' \in \mathbb{F}_{q^m}^n$ be multipliers. Let k, k' be two positive integers, then*

$$\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \star \mathbf{GRS}_{k'}(\boldsymbol{x}, \boldsymbol{y}') = \mathbf{GRS}_{k+k'-1}(\boldsymbol{x}, \boldsymbol{y} \star \boldsymbol{y}').$$

Proof. See for instance [10, Proposition 6]. □

3 Conductors

In this section, we introduce a fundamental object in the attack to follow. This object was already used in [11, 13] without being named. We chose here to call it *conductor*. The rationale behind this terminology is explained in Remark 2.

Definition 20. Let $\mathscr{C}$ and $\mathscr{D}$ be two codes of length n over $\mathbb{F}_q$. The *conductor of $\mathscr{D}$ into $\mathscr{C}$* is defined as the largest code $\mathscr{Z} \subseteq \mathbb{F}_q^n$ such that $\mathscr{D} \star \mathscr{Z} \subseteq \mathscr{C}$. That is:

$$\mathbf{Cond}(\mathscr{D}, \mathscr{C}) \stackrel{\text{def}}{=} \{\boldsymbol{u} \in \mathbb{F}_q^n \mid \boldsymbol{u} \star \mathscr{D} \subseteq \mathscr{C}\}.$$

Proposition 21. *Let $\mathscr{D}, \mathscr{C} \subseteq \mathbb{F}_q^n$ be two codes, then*

$$\mathbf{Cond}(\mathscr{D}, \mathscr{C}) = \left(\mathscr{D} \star \mathscr{C}^\perp\right)^\perp.$$

Proof. See [11, 13]. □

Remark 2. The terminology *conductor* has been borrowed from number theory in which the conductor of two subrings $\mathcal{O}, \mathcal{O}'$ of the ring of integers $\mathcal{O}_K$ of a number field K is the largest ideal $\mathfrak{P}$ of $\mathcal{O}_K$ such that $\mathfrak{P} \cdot \mathcal{O} \subseteq \mathcal{O}'$.

3.1 Conductors of GRS codes

Before discussing the behaviour of conductors of alternant codes, let us start with GRS codes.

Proposition 22. *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_{q^m}^n$ be a support and a multiplier. Let $k \leqslant k'$ be two integers less than n. Then,*

$$\mathbf{Cond}(\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}), \mathbf{GRS}_{k'}(\boldsymbol{x}, \boldsymbol{y})) = \mathbf{RS}_{k'-k+1}(\boldsymbol{x}).$$

Proof. Let $\mathscr{E}$ denote the conductor. From Proposition 21 and Lemma 7,

$$\mathscr{E}^\perp = \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \star \mathbf{GRS}_{n-k'}(\boldsymbol{x}, \boldsymbol{y}^\perp) = \mathbf{GRS}_{n-k'+k-1}(\boldsymbol{x}, \boldsymbol{y} \star \boldsymbol{y}^\perp).$$

Note that

$$\boldsymbol{y} \star \boldsymbol{y}^\perp = \left(\frac{1}{\pi'_{\boldsymbol{x}}(x_1)}, \ldots, \frac{1}{\pi'_{\boldsymbol{x}}(x_n)}\right).$$

Then, using Lemma 7 again, we get

$$\mathscr{E} = \mathbf{GRS}_{k'-k+1}(\boldsymbol{x}, (\boldsymbol{y} \star \boldsymbol{y}^\perp)^\perp) = \mathbf{RS}_{k'-k+1}(\boldsymbol{x}).$$

□

Let us emphasize a very interesting aspect of Proposition 21. We considered the conductor of a GRS code into another one having the same support and multiplier. The point is that the conductor **does not depend on $\boldsymbol{y}$**. Hence the computation of a conductor permits to get rid of the multiplier and to obtain a much easier code to study : a Reed–Solomon code.

As we see further, when dealing with alternant codes, instead of getting a Reed–Solomon code, the computation of some conductor provides a particular subfield subcode of a Reed–Solomon code, which we called the *norm–trace code*. The next subsection is devoted to the study of this object.

3.2 The norm–trace code

Notation 6. In what follows, we fix $\alpha \in \mathbb{F}_{q^2}$ such that $\mathrm{Tr}(\alpha) = 1$. In particular, $(1, \alpha)$ form an $\mathbb{F}_q$–basis of $\mathbb{F}_{q^2}$.

Definition 23 (Norm trace code)**.** Let $\boldsymbol{x} \in \mathbb{F}_{q^2}^n$ be a support. The *norm–trace code* $\mathscr{NT}(\boldsymbol{x}) \subseteq \mathbb{F}_q^n$ is defined as

$$\mathscr{NT}(\boldsymbol{x}) \stackrel{\text{def}}{=} \langle \mathbf{1}, \mathrm{Tr}(\boldsymbol{x}), \mathrm{Tr}(\alpha\boldsymbol{x}), \mathrm{N}(\boldsymbol{x}) \rangle_{\mathbb{F}_q}.$$

The code $\mathscr{NT}(\boldsymbol{x})$ turns out to be a very peculiar alternant code since it is a subfield subcode of a Reed–Solomon code.

Proposition 24. *Let $\boldsymbol{x} \in \mathbb{F}_{q^2}^n$ be a support and $n = q^2$. Then, for any $q+1 < k < 2q+1$, we have*

$$\mathscr{NT}(\boldsymbol{x}) = \mathbf{RS}_k(\boldsymbol{x}) \cap \mathbb{F}_q^n.$$

Proof. The case $k = q+2$ is proved in [13, Proposition 33]. The proof of the general case is very similar. We give it for the sake of convenience.

Inclusion "$\subseteq$" is obvious. Let us discuss the converse inclusion. The goal is to describe polynomials $h \in \mathbb{F}_{q^2}[z]_{<k}$ such that

$$\forall x \in \mathbb{F}_{q^2},\ h(x) = h(x)^q.$$

This is equivalent to

$$h \equiv h^q \mod (z^{q^2} - z).$$

Writing, $h(z) = \sum_{i=0}^{k-1} h_i z^i$, this gives the system

$$\left\{ \begin{array}{lcll} h_0 & = & h_0^q & \\ h_q & = & h_1^q & \\ h_{q+1} & = & h_{q+1}^q & \\ h_i & = & 0, & \forall i \in \{2, \ldots, k-1\} \setminus \{q, q+1\}. \end{array} \right.$$

We deduce an $\mathbb{F}_q$–basis of polynomials $1, z^q + z, \alpha^q z^q + \alpha z, z^{q+1}$, which proves the result. □

We finish this subsection with a heuristic extending Proposition 24 to the case where $n < q^2$. This heuristic is already discussed in [13], where computer aided experiments providing evidences for it are given.

Heuristic 25. *Let $\boldsymbol{x} \in \mathbb{F}_{q^2}^n$ be a support and $n \geqslant 4q$. Then, for any $q+1 < k < 2q+1$, we have*

$$\mathscr{NT}(\boldsymbol{x}) = \mathbf{RS}_k(\boldsymbol{x}) \cap \mathbb{F}_q^n.$$

3.3 Conductor of some alternant codes

When dealing with alternant codes, proving equalities becomes difficult. We can at least prove the following theorem.

Theorem 26. *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_{q^2}^n$ be a support and a multiplier. Let $r' \geqslant r$ be two positive integers. Then,*

$$\mathbf{RS}_{r'-r+1}(\boldsymbol{x}) \cap \mathbb{F}_q^n \subseteq \mathbf{Cond}(\mathscr{A}_{r'}(\boldsymbol{x}, \boldsymbol{y}), \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})).$$

Proof. Consider the Schur product

$$\begin{aligned}\left(\mathbf{RS}_{r'-r+1}(\boldsymbol{x}) \cap \mathbb{F}_q^n\right) \star \mathscr{A}_{r'}(\boldsymbol{x}, \boldsymbol{y}) & \\ &= \left(\mathbf{RS}_{r'-r+1}(\boldsymbol{x}) \cap \mathbb{F}_q^n\right) \star \left(\mathbf{GRS}_{n-r'}(\boldsymbol{x}, \boldsymbol{y}^\perp) \cap \mathbb{F}_q^n\right) \\ &\subseteq \left(\mathbf{RS}_{r'-r+1}(\boldsymbol{x}) \star \mathbf{GRS}_{n-r'}(\boldsymbol{x}, \boldsymbol{y}^\perp)\right) \cap \mathbb{F}_q^n.\end{aligned}$$

Next, using Theorem 19,

$$\begin{aligned}\left(\mathbf{RS}_{r'-r+1}(\boldsymbol{x}) \cap \mathbb{F}_q^n\right) \star \mathscr{A}_{r'}(\boldsymbol{x}, \boldsymbol{y}) &\subseteq \mathbf{GRS}_{n-r}(\boldsymbol{x}, \boldsymbol{y}^\perp) \cap \mathbb{F}_q^n \\ &\subseteq \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}).\end{aligned}$$

The last inclusion is a direct consequence of Lemma 7 and Definition 5. □

Theorem 26 is not used as stated in the attack, we actually use a more general heuristic which is discussed in the sequel.

Heuristic 27. *In the context of Theorem 26. Suppose that $n \geqslant 4q$ and $q < r - r' < 2q$. Let $\mathscr{D}$ be a subcode of $\mathscr{A}_{r'}(\boldsymbol{x}, \boldsymbol{y})$ such that*

(i) $\dim \mathscr{D} \cdot \dim \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp \geqslant n$;

(ii) $\mathscr{D} \not\subset \mathscr{A}_{r'+1}(\boldsymbol{x}, \boldsymbol{y})$;

(iii) a generator matrix of $\mathscr{D}$ has no zero column.

Then,

$$\mathbf{Cond}(\mathscr{D}, \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})) = \mathscr{NT}(\boldsymbol{x}).$$

Let us give some evidences for this heuristic. From Proposition 21,

$$\mathbf{Cond}(\mathscr{D}, \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})) = \left(\mathscr{D} \star \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp\right)^\perp.$$

Next, from (2), we have

$$\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp = \mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}(\mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})).$$

Since $\mathscr{D}$ is a code over $\mathbb{F}_q$ and by the $\mathbb{F}_q$–linearity of the trace map, we get

$$\mathscr{D} \star \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp = \mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}\left(\mathscr{D} \star \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})\right).$$

Now, we use the fact that $\mathscr{D}$ is contained in $\mathscr{A}_{r'}(\boldsymbol{x}, \boldsymbol{y})$ and hence, from (1) is a subset of a GRS code. Namely,

$$\mathscr{D} \subset \mathbf{GRS}_{n-r'}(\boldsymbol{x}, \boldsymbol{y}^\perp), \quad \text{where} \quad \boldsymbol{y}^\perp = \left(\frac{1}{\pi'_{\boldsymbol{x}}(x_1)y_1}, \ldots, \frac{1}{\pi'_{\boldsymbol{x}}(x_n)y_n}\right).$$

Therefore, thanks to Theorem 19, we get

$$\mathscr{D} \star \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp \subseteq \mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}\left(\mathbf{GRS}_{n-r'+r-1}(\boldsymbol{x}, \boldsymbol{y} \star \boldsymbol{y}^\perp)\right). \tag{4}$$

Here, let us note that $\mathscr{D} \star \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp$ is spanned by $\dim \mathscr{D} \cdot \dim \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp$ generators which are obtained by computing the Schur products of elements of a basis of $\mathscr{D}$ by elements of a basis of $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp$. By (i), the number of such generators exceeds n. For this reason, it is very reasonable to hope that this Schur product will fill in the target code and hence that actually,

$$\mathscr{D} \star \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp = \mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}\left(\mathbf{GRS}_{n-r'+r-1}(\boldsymbol{x}, \boldsymbol{y} \star \boldsymbol{y}^\perp)\right).$$

Next, we have

$$\boldsymbol{y} \star \boldsymbol{y}^\perp = \left(\frac{1}{\pi'_{\boldsymbol{x}}(x_1)}, \ldots, \frac{1}{\pi'_{\boldsymbol{x}}(x_n)}\right).$$

Therefore, using Lemma 7, we conclude that

$$\left(\mathscr{D} \star \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp\right)^\perp = \mathbf{RS}_{r'-r+1}(\boldsymbol{x}) \cap \mathbb{F}_q^n.$$

Using Heuristic 25, we get the result.

Remark 3. Assumption (ii) permits to avoid the situation where the conductor could be the subfield subcode of a larger Reed–Solomon code. Assumption (iii) permits to avoid the presence of words of weight 1 in the conductor which would not be elements of a Reed–Solomon code.

Further discussion on the Heuristic In all our computer experiments, we never observed any phenomenon contradicting this heuristic.

4 Presentation of the attack

4.1 Context

Remind that the extension degree is always $m = 2$. The public code is the QD alternant code

$$\mathscr{C}_{\mathrm{pub}} \stackrel{\mathrm{def}}{=} \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}),$$

with a permutation group $\mathcal{G}$ of cardinality $|\mathcal{G}| = 2^\gamma$. We remind the parameters listed in § 1.7. The code has a length $n = n_0 2^\gamma$, dimension k and is defined over a field $\mathbb{F}_q$ and $q = 2^\ell$ for some positive integer ℓ. The degree r of the alternant code is also a multiple of $|\mathcal{G}| = 2^\gamma$ and hence is of the form $r = r_0 2^\gamma$. We suppose from now on that the lower bound on the dimension k given by Proposition 6 is reached. Namely that $k = n - 2r$. This always holds in the parameters proposed in [3]. We finally set $k_0 = k/2^\gamma$. In summary, we have the following notation

$$n = n_0 2^\gamma, \quad k = k_0 2^\gamma, \quad r = r_0 2^\gamma. \tag{5}$$

4.2 Description of the attack

In § 4.4, we introduce a subcode $\mathscr{D}$ of codimension $\frac{2q}{|\mathcal{G}|}$ of $(\mathscr{C}_{\rm pub})^{\mathcal{G}}$. This subcode $\mathscr{D}$ is unknown, while its knowledge of $\mathscr{D}$ permits to recover $\mathscr{NT}(\boldsymbol{x})$ as the conductor $\mathbf{Cond}(\mathscr{D}, \mathscr{C}_{\rm pub})$. The difficult part of the attack consists in guessing this unknown code $\mathscr{D}$.

The attack is can be summarised as follows:

(1) Compute $(\mathscr{C}_{\rm pub})^{\mathcal{G}}$;

(2) Guess the subcode $\mathscr{D}$ of $(\mathscr{C}_{\rm pub})^{\mathcal{G}}$ of codimension $\frac{2q}{|\mathcal{G}|}$ such that

$$\mathbf{Cond}(\mathscr{D}, \mathscr{C}_{\rm pub}) = \mathscr{NT}(\boldsymbol{x});$$

(3) Determine $\boldsymbol{x}$ from $\mathscr{NT}(\boldsymbol{x})$ and then $\boldsymbol{y}$ from $\boldsymbol{x}$.

The difficult part is clearly the second one : how to guess $\mathscr{D}$? We present two manners to realise this guess.

- The first one consists in performing exhaustive search on subcodes of codimension $\frac{2q}{|\mathcal{G}|}$ of $(\mathscr{C}_{\rm pub})^{\mathcal{G}}$.
- The second one consists in finding both $\mathscr{D}$ and $\mathscr{NT}(\boldsymbol{x})$ by solving a bilinear system using Gröbner bases.

The first approach has an important cost but which remains significantly below the expected security level of DAGS proposed parameters. For the second approach, we did not succeed to get a relevant estimate of the work factor but its practical implementation permits to break `DAGS_1` in about 20 minutes and `DAGS_5` in less than one minute (see § 6 for further details on the implementation). We did not succeed to break `DAGS_3` parameters using the second approach. On the other hand the first approach would have a work factor of $\approx 2^{80}$ for keys with an expected security of 192 bits.

The remainder of this section is devoted to detail the different aspects of the attack.

4.3 Fundamental degree properties of the invariant subcode

A crucial statement for the attack is:

Theorem 28. *Let s be an integer of the form $s = 2^{\gamma}s_0$. Suppose that $\mathscr{A}_{s_0}(\overline{\psi_{\mathcal{G}}(\boldsymbol{x})}, \overline{\boldsymbol{y}})$ is fully non degenerated (see Definition 8 and § 1.5 for notation $\psi_{\mathcal{G}}$, $\overline{\boldsymbol{y}}$ and so on). Then,*

(a) $\mathscr{A}_s(\boldsymbol{x}, \boldsymbol{y})^{\mathcal{G}} \subseteq \mathscr{A}_{s+|\mathcal{G}|-1}(\boldsymbol{x}, \boldsymbol{y})$;

(b) $\mathscr{A}_s(\boldsymbol{x}, \boldsymbol{y})^{\mathcal{G}} \not\subseteq \mathscr{A}_{s+|\mathcal{G}|}(\boldsymbol{x}, \boldsymbol{y})$.

Remark 4. Note that the sequence of alternant codes for increasing degrees is decreasing:

$$\mathscr{A}_s(\boldsymbol{x}, \boldsymbol{y}) \supseteq \mathscr{A}_{s+1}(\boldsymbol{x}, \boldsymbol{y}) \supseteq \cdots \supseteq \mathscr{A}_{s+|\mathcal{G}|-1}(\boldsymbol{x}, \boldsymbol{y}) \supseteq \mathscr{A}_{s+|\mathcal{G}|}(\boldsymbol{x}, \boldsymbol{y}).$$

Hence, the invariant code is contained in a smaller alternant code, corresponding to the evaluation of polynomials of lower degree.

Proof. From (1), we have

$$\mathscr{A}_s(\boldsymbol{x},\boldsymbol{y}) = \left\{ \left(\frac{1}{y_i \pi'_{\boldsymbol{x}}(x_i)} f(x_i) \right)_{i=1,\dots,n} \;\middle|\; f \in \mathbb{F}_{q^2}[z]_{<n-s} \right\} \cap \mathbb{F}_q^n.$$

This code is obtained by evaluation of polynomials of degree up to

$$n - s - 1 = (2^\gamma(n_0 - s_0) - 1).$$

Next, from Theorem 17, the invariant codewords of $\mathscr{A}_s(\boldsymbol{x},\boldsymbol{y})$ come from evaluations of polynomials of the form $h \circ \psi_{\mathcal{G}}$. Such polynomials have a degree which is a multiple of $\deg \psi_{\mathcal{G}} = 2^\gamma$ and hence their degree cannot exceed $2^\gamma(n_0 - s_0 - 1)$. Thus, they should lie in $\mathbb{F}_{q^2}[z]_{\leqslant n-s-|\mathcal{G}|} = \mathbb{F}_{q^2}[z]_{<n-s-|\mathcal{G}|+1}$. This leads to

$$\begin{aligned}\mathscr{A}_s(\boldsymbol{x},\boldsymbol{y})^{\mathcal{G}} &\subseteq \left\{ \left(\frac{1}{y_i \pi'_{\boldsymbol{x}}(x_i)} f(x_i) \right)_{i=1,\dots,n} \;\middle|\; f \in \mathbb{F}_{q^2}[z]_{<n-s-|\mathcal{G}|+1} \right\} \cap \mathbb{F}_q^n \\ &\subseteq \mathscr{A}_{s+|\mathcal{G}|-1}(\boldsymbol{x},\boldsymbol{y}).\end{aligned}$$

This proves (a).

To prove (b), note that the assumption on $\mathscr{A}_{s_0}(\overline{\psi_{\mathcal{G}}(\boldsymbol{x})}, \overline{\boldsymbol{y}})$ asserts the existence of $f \in \mathbb{F}_{q^2}[z]_{<n_0-s_0}$ such that $\deg f = n_0 - s_0 - 1$ and $f(\overline{\psi_{\mathcal{G}}(\boldsymbol{x})}) \in \mathbb{F}_q^{n_0}$. Thus, $f(\psi_{\mathcal{G}}(\boldsymbol{x})) \in \mathbb{F}_q^n$ and $\deg(f \circ \psi_{\mathcal{G}}) = n - s - |\mathcal{G}|$. Therefore $f(\psi(\boldsymbol{x})) \in \mathscr{A}_s(\boldsymbol{x},\boldsymbol{y})^{\mathcal{G}}$ and $\mathscr{A}_s(\boldsymbol{x},\boldsymbol{y})^{\mathcal{G}}$ contains an element of $\mathscr{A}_{s+|\mathcal{G}|-1}(\boldsymbol{x},\boldsymbol{y})$ which is not in $\mathscr{A}_{s+|\mathcal{G}|}(\boldsymbol{x},\boldsymbol{y})$. □

4.4 The subcode $\mathscr{D}$

Definition 29. Suppose that $|\mathcal{G}| \leqslant q$. We define the code $\mathscr{D}$ as

$$\mathscr{D} \stackrel{\text{def}}{=} \mathscr{A}_{r+q}(\boldsymbol{x},\boldsymbol{y})^{\mathcal{G}}.$$

Remark 5. For parameters suggested in DAGS, we always have $|\mathcal{G}| \leqslant q$, with strict equality for `DAGS_1` and `DAGS_3` and equality for `DAGS_5`.

Remark 6. The case $q < |\mathcal{G}|$ which never holds in DAGS suggested parameters would be particularly easy to treat. In such a situation, replacing possibly $\mathcal{G}$ by a subgroup, one can suppose that $|\mathcal{G}| = 2q$. Next, according to Theorem 28, and Heuristic 27, we have

$$\mathbf{Cond}((\mathscr{C}_{\text{pub}})^{\mathcal{G}}, \mathscr{C}_{\text{pub}}) = \mathscr{NT}(\boldsymbol{x}),$$

which would provide a very simple manner to compute $\mathscr{NT}(\boldsymbol{x})$.

The following result is the key of the attack.

Theorem 30. *Under Heuristics 25 and 27 and assuming that* $\overline{\mathscr{A}_{r+q}(\boldsymbol{x},\boldsymbol{y})}^{\mathcal{G}}$ *is fully non degenerated (see Definition 8), we have*

$$\mathbf{Cond}(\mathscr{D}, \mathscr{C}_{pub}) = \mathscr{NT}(\boldsymbol{x}).$$

Proof. It is a direct consequence of Theorem 28 and Heuristic 27. □

Proposition 31. *The code* $\mathscr{D}$ *has codimension* $\leqslant \frac{2q}{|\mathcal{G}|} = 2^{\ell-\gamma+1}$ *in* $(\mathscr{C}_{pub})^{\mathcal{G}}$.

Proof. Using Theorem 18, we know that $\mathscr{D}$ has the same dimension as $\mathscr{A}_{r_0+\frac{q}{|\mathcal{G}|}}(\overline{\psi_{\mathcal{G}}(\boldsymbol{x})}, \overline{\boldsymbol{y}})$. This code has dimension $\geqslant n_0 - 2(r_0 + \frac{q}{|\mathcal{G}|})$. Since $\dim (\mathscr{C}_{\rm pub})^{\mathcal{G}} = k_0 = n_0 - 2r_0$, we get the result. □

Remark 7. Actually the codimension equals $2^{\ell-\gamma+1}$ almost all the time.

Example 1.
- For `DAGS_1`, $\mathscr{D} = \mathscr{A}_{240}(\boldsymbol{x}, \boldsymbol{y})^{\mathcal{G}}$ and the code has codimension 4 in $(\mathscr{C}_{\rm pub})^{\mathcal{G}}$;
- For `DAGS_3`, $\mathscr{D} = \mathscr{A}_{416}(\boldsymbol{x}, \boldsymbol{y})^{\mathcal{G}}$ and the code has codimension 4 in $(\mathscr{C}_{\rm pub})^{\mathcal{G}}$;
- For `DAGS_5`, $\mathscr{D} = \mathscr{A}_{768}(\boldsymbol{x}, \boldsymbol{y})^{\mathcal{G}}$ and the code has codimension 2 in $(\mathscr{C}_{\rm pub})^{\mathcal{G}}$;

When starting the attack, the code $\mathscr{D}$ is unknown. In the sequel, we present two manners to recover it.

4.5 First approach, brute force search of $\mathscr{D}$

A first way of getting $\mathscr{D}$ and then of obtaining $\mathscr{NT}(\boldsymbol{x})$ consists in enumerating all the subspaces $\mathscr{X} \subseteq (\mathscr{C}_{\rm pub})^{\mathcal{G}}$ of codimension $\frac{2q}{|\mathcal{G}|}$ until we find one such that $\mathbf{Cond}(\mathscr{X}, \mathscr{C}_{\rm pub})$ has dimension 4. Indeed, for an arbitrary $\mathscr{X}$ the conductor will have dimension 1 and be generated by $\mathbf{1}$, while for $\mathscr{X} = \mathscr{D}$ the conductor will be $\mathscr{NT}(\boldsymbol{x})$ which has dimension 4.

The number of subspaces to enumerate is in $O(q^{2(2q/|\mathcal{G}|)(k_0-2q/|\mathcal{G}|)})$ which is in general much too large to make the attack practical. It is however possible to reduce the cost of brute force attack as follows.

4.5.1 Using random subcodes of dimension 2

For any parameter set proposed in DAGS, the public code has a rate k/n less than $1/2$. Hence, its dual has rate larger than $1/2$. Therefore, according to Heuristic 27, given a random subcode $\mathscr{D}_0$ of $\mathscr{D}$ of dimension 2, then $\mathbf{Cond}(\mathscr{D}_0, \mathscr{C}_{\rm pub}) = \mathscr{NT}(\boldsymbol{x})$ with a high probability.

Thus, one can proceed as follows

- Pick two independent vectors $\boldsymbol{c}, \boldsymbol{c}' \in (\mathscr{C}_{\rm pub})^{\mathcal{G}}$ at random and compute $\mathbf{Cond}(\langle \boldsymbol{c}, \boldsymbol{c}' \rangle, \mathscr{C}_{\rm pub})$;
- If the conductor has dimension 4, you probably found $\mathscr{NT}(\boldsymbol{x})$, then pursue the attack as explained in § 4.7.
- Else, try again.

The probability that $\boldsymbol{c}, \boldsymbol{c}' \in \mathscr{D}$ equals $q^{-\frac{4q}{|\mathcal{G}|}}$. Therefore, one may have found $\mathscr{NT}(\boldsymbol{x})$ after $O(q^{\frac{4q}{|\mathcal{G}|}})$ computations of conductors.

Example 2. The average number of computations of conductors will be

- $O(q^8) = O(2^{40})$ for `DAGS_1`;
- $O(q^8) = O(2^{48})$ for `DAGS_3`;
- $O(q^4) = O(2^{24})$ for `DAGS_5`.

4.5.2 Using shortened codes

Another manner consists in replacing the public code by one of its shortenings. For that, we shorten $\mathscr{C}_{\rm pub} = \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ at a set of $a = a_0 2^\gamma$ positions which is a union of blocks, so that the shortened code remains QD. We choose the integer a such that the invariant subcode of the shortened code has dimension $2 + \frac{2q}{|\mathcal{G}|}$ and hence the shortening of $\mathscr{D}$ has dimension 2. Let $\mathcal{I}$ be such a subset of positions. To determine $\mathcal{S}_{\mathcal{I}}(\mathscr{D})$, we can enumerate any subspace $\mathscr{X}$ of dimension 2 of $\mathcal{S}_{\mathcal{I}}(\mathscr{C}_{\rm pub})$ and compute $\mathbf{Cond}(\mathscr{X}, \mathcal{S}_{\mathcal{I}}(\mathscr{C}_{\rm pub}))$. In general, we get the trivial code spanned by the all–one codeword $\mathbf{1}$. If the conductor has dimension 4 it is highly likely that we found $\mathcal{S}_{\mathcal{I}}(\mathscr{D})$ and that the computed conductor equals $\mathscr{NT}(\boldsymbol{x}_{\mathcal{I}})$.

The number of such spaces we enumerate is in $O(q^{\frac{4q}{|\mathcal{G}|}})$. Hence the average number of conductors we have to compute is in $O(q^{\frac{4q}{|\mathcal{G}|}})$, which is very similar to the cost of the previous method based on random subcodes of dimension 2 in § 4.5.1.

4.6 Second approach, solving a bilinear system

An alternative approach to recover $\mathscr{D}$ and $\mathscr{NT}(\boldsymbol{x})$ consists in solving a bilinear system. The idea is the following one. Since $\mathrm{Tr}(\boldsymbol{x}) \in \mathbf{Cond}(\mathscr{D}, \mathscr{C}_{\rm pub})$ and, from Proposition 21, $\mathbf{Cond}(\mathscr{D}, \mathscr{C}_{\rm pub}) = (\mathscr{D} \star {\mathscr{C}_{\rm pub}}^\perp)^\perp$, then

$$\boldsymbol{G}_{\mathscr{D}\star{\mathscr{C}_{\rm pub}}^\perp} \cdot \mathrm{Tr}(\boldsymbol{x})^\top = 0,$$

where $\boldsymbol{G}_{\mathscr{D}\star{\mathscr{C}_{\rm pub}}^\perp}$ denotes a generator matrix of $\mathscr{D}\star{\mathscr{C}_{\rm pub}}^\perp$. The above identity holds true when replacing $\mathrm{Tr}(\boldsymbol{x})$ by $\mathrm{Tr}(\beta\boldsymbol{x})$ for any $\beta \in \mathbb{F}_{q^2}$. Hence,

$$\boldsymbol{G}_{\mathscr{D}\star{\mathscr{C}_{\rm pub}}^\perp} \cdot \boldsymbol{x}^\top = 0. \tag{6}$$

The above identity provides the system we wish to solve. We have two type of unknowns : the code $\mathscr{D}$ and the support vector $\boldsymbol{x}$. Set $c \stackrel{\rm def}{=} \frac{2q}{|\mathcal{G}|}$ the codimension of $\mathscr{D}$ in $(\mathscr{C}_{\rm pub})^{\mathcal{G}}$. For $\mathscr{D}$, let us introduce $(k_0 - c)k_0$ formal variables $U_{11}, \ldots, U_{1,c}, \ldots, U_{k_0-c,1}, \ldots, U_{k_0-c,c}$ and set

$$\boldsymbol{U} \stackrel{\rm def}{=} \begin{pmatrix} U_{11} & \cdots & U_{1,c} \\ \vdots & & \vdots \\ U_{k_0-c,1} & \cdots & U_{k_0-c,c} \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{G}(U_{ij}) \stackrel{\rm def}{=} \left(\boldsymbol{I}_{k_0-c} \mid \boldsymbol{U} \right) \cdot \boldsymbol{G}^{\rm inv},$$

where $\boldsymbol{I}_{k_0-c}$ denotes the $(k_0-c)\times(k_0-c)$ identity matrix and $\boldsymbol{G}^{\rm inv}$ denotes a $k_0 \times n_0$ generator matrix of $(\mathscr{C}_{\rm pub})^{\mathcal{G}}$. It is probable that $\mathscr{D}$ has a generator matrix of the form $\boldsymbol{G}(u_{ij})$ for some special values $u_{11}, \ldots, u_{k_0-c,c} \in \mathbb{F}_q$. The case where $\mathscr{D}$ has no generator matrix of this form is rare and can be addressed by choosing another generator matrix for $(\mathscr{C}_{\rm pub})^{\mathcal{G}}$.

Now, let $\boldsymbol{H}$ be a parity–check matrix of $\mathscr{C}_{\rm pub}$. A generator matrix of $\mathscr{D} \star {\mathscr{C}_{\rm pub}}^\perp$ can be obtained by constructing a matrix whose rows list all the possible Schur products of one row of a generator matrix of $\mathscr{D}$ by one row of a parity–check matrix of $\mathscr{C}_{\rm pub}$. Therefore, let $\boldsymbol{R}(U_{ij})$ be a matrix with entries in $\mathbb{F}_q[U_{1,1}, \ldots, U_{k_0-c,c}]$ whose rows list all the possible Schur products of one row of $\boldsymbol{G}(U_{i,j})$ and one row of $\boldsymbol{H}$. Hence, there is a specialisation $u_{11}, \ldots, u_{k_0-c,c} \in \mathbb{F}_q$ of the variables U_{ij} such that $\boldsymbol{R}(u_{ij})$ is a generator matrix of $\mathscr{D} \star {\mathscr{C}_{\rm pub}}^\perp$.

The second set of variables $X_1, \ldots, X_n$ corresponds to the entries of $\boldsymbol{x}$. Using (6), the bilinear system we have to solve is nothing but

$$\boldsymbol{R}(U_{ij}) \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} = 0. \tag{7}$$

4.6.1 Reducing the number of variables

Actually, it is possible to reduce the number of variables using three different tricks.

1. Since the code is QD, the vector $\boldsymbol{x}$ is a union of orbits under the action of the additive group $\mathcal{G}$. Therefore, one can introduce formal variables $A_1, \ldots, A_\gamma$ corresponding to the generators of $\mathcal{G}$. Then, one can replace $(X_1, \ldots, X_n)$ by

$$(T_1,\ T_1 + A_1,\ \ldots\ ,\ T_1 + A_1 + \cdots + A_\gamma,\ T_2, T_2 + A_1,\ \ldots\ ). \tag{8}$$

   for some variables $T_1, \ldots, T_{n_0}$.

2. Without loss of generality and because of the 2–transitive action of the affine group on $\mathbb{F}_{q^2}$, one can suppose that the first entries of $\boldsymbol{x}$ are 0 and 1 respectively (see for instance [13, Appendix A]). Therefore, in (8), one can replace T_1 by 0 and A_1 by 1.

3. Similarly to the approach of § 4.5, one can shorten the codes so that $\mathscr{D}$ has only dimension 2, which reduces the number of variables U_{ij} to $2c$ and also reduces the length of the support we seek and hence reduces the number of the variables T_i.

4.7 Finishing the attack

When the previous step of the attack is over, then, if we used the first approach based on a brute force search of $\mathscr{D}$, we know at least $\mathscr{NT}(\boldsymbol{x})$ or $\mathscr{NT}(\boldsymbol{x}_\mathcal{I})$ for some set $\mathcal{I}$ of positions. If we used the second approach, then $\boldsymbol{x}$ is already computed, or at least $\boldsymbol{x}_\mathcal{I}$ for some set of indexes $\mathcal{I}$. Thus, there remains to be able to

(1) recover $\boldsymbol{x}$ from $\mathscr{NT}(\boldsymbol{x})$ or $\boldsymbol{x}_\mathcal{I}$ from $\mathscr{NT}(\boldsymbol{x}_\mathcal{I})$;

(2) recover $\boldsymbol{y}$ from $\boldsymbol{x}$ or $\boldsymbol{y}_\mathcal{I}$ from $\boldsymbol{x}_\mathcal{I}$;

(3) recover $\boldsymbol{x}, \boldsymbol{y}$ from the knowledge of $\boldsymbol{x}_\mathcal{I}, \boldsymbol{y}_\mathcal{I}$.

Let us treat these three questions.

4.7.1 Recovering $\boldsymbol{x}$ from $\mathscr{NT}(\boldsymbol{x})$

Remind that the code $\mathscr{NT}(\boldsymbol{x})$ has dimension 4 over $\mathbb{F}_q$ and is spanned by $\mathbf{1}, \mathrm{Tr}(\boldsymbol{x}), \mathrm{Tr}(\alpha\boldsymbol{x}), \mathrm{N}(\boldsymbol{x})$. It is not difficult to prove that the same code after base field extension satisfies

$$\mathscr{NT}(\boldsymbol{x}) \otimes \mathbb{F}_{q^2} = \langle \mathbf{1}, \boldsymbol{x}, \boldsymbol{x}^{\star q}, \boldsymbol{x}^{\star(q+1)} \rangle.$$

This code is peculiar in the sense that its square is smaller than the square of an arbitrary code of dimension 4. Indeed, according to [9], the square of a random code of dimension 4 has dimension 10 with a high probability, while:

Lemma 32. *If $n > 2q+2$, then* $\dim \mathscr{NT}(\boldsymbol{x})^{\star 2} = 9$.

Proof. Note first that

$$\dim_{\mathbb{F}_q} \mathscr{NT}(\boldsymbol{x})^{\star 2} = \dim_{\mathbb{F}_{q^2}} (\mathscr{NT}(\boldsymbol{x})^{\star 2}) \otimes \mathbb{F}_{q^2} = \dim_{\mathbb{F}_{q^2}} (\mathscr{NT}(\boldsymbol{x}) \otimes \mathbb{F}_{q^2})^{\star 2}.$$

Hence, let us study the square of $\mathscr{NT}(\boldsymbol{x}) \otimes \mathbb{F}_{q^2}$,

$$\langle \mathbf{1}, \boldsymbol{x}, \boldsymbol{x}^{\star q}, \boldsymbol{x}^{\star (q+1)} \rangle^{\star 2} = \\ \langle \mathbf{1}, \boldsymbol{x}, \boldsymbol{x}^{\star 2}, \boldsymbol{x}^{\star q}, \boldsymbol{x}^{\star (q+1)}, \boldsymbol{x}^{\star (q+2)}, \boldsymbol{x}^{\star (2q)}, \boldsymbol{x}^{\star (2q+1)}, \boldsymbol{x}^{\star (2q+2)} \rangle.$$

One can check that these vectors are independent when $n > 2q+2$. □

Remark 8. The previous lemma provides an additional test to check whether the code we computed was actually $\mathscr{NT}(\boldsymbol{x})$ by simply computing its square.

Because of the 2–transitivity of the affine group on $\mathbb{F}_{q^2}$, without loss of generality, one can suppose that the first entry of $\boldsymbol{x}$ is 0 and the second one is 1 (see for instance [13, Appendix A]). Therefore, after shortening $\mathscr{NT}(\boldsymbol{x}) \otimes \mathbb{F}_{q^2}$ we get a code that we call $\mathscr{S}$, which is of the form

$$\mathscr{S} \stackrel{\text{def}}{=} \mathcal{S}_{\{1\}} \left(\mathscr{NT}(\boldsymbol{x}) \otimes \mathbb{F}_{q^2} \right) = \langle \boldsymbol{x}, \boldsymbol{x}^{\star q}, \boldsymbol{x}^{\star (q+1)} \rangle_{\mathbb{F}_{q^2}}.$$

Next, a simple calculation shows that

$$\mathscr{S} \cap \mathscr{S}^{\star 2} = \langle \boldsymbol{x}^{\star (q+1)} \rangle.$$

Since, the second entry of $\boldsymbol{x}$ has been set to 1, we can deduce the value of $\boldsymbol{x}^{\star (q+1)}$.

Remark 9. Actually, both $\mathscr{S}$ and $\mathscr{NT}(\boldsymbol{x})$ have a basis defined over $\mathbb{F}_q$, therefore, to get $\langle \boldsymbol{x}^{\star (q+1)} \rangle_{\mathbb{F}_q}$ it is sufficient to perform any computation on codes defined over $\mathbb{F}_q$.

Now, finding $\boldsymbol{x}$ is easy: enumerate the affine subspace of $\mathscr{NT}(\boldsymbol{x}) \otimes \mathbb{F}_{q^2}$ of vectors whose first entry is 0 and second entry is 1 (or equivalently, the affine subspace of vectors of $\mathscr{S}$ whose first entry equals 1). For any such vector $\boldsymbol{c}$, compute $\boldsymbol{c}^{\star (q+1)}$. If $\boldsymbol{c}^{\star (q+1)} = \boldsymbol{x}^{\star (q+1)}$, then $\boldsymbol{c}$ equals either $\boldsymbol{x}$ or $\boldsymbol{x}^{\star q}$. Since $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) = \mathscr{A}_r(\boldsymbol{x}^{\star q}, \boldsymbol{y}^{\star q})$ (see for instance [13, Lemma 39]), taking $\boldsymbol{x}$ or $\boldsymbol{x}^{\star q}$ has no importance. Thus, without loss of generality, one can suppose $\boldsymbol{x}$ has been found.

4.7.2 Recovering $\boldsymbol{y}$ from $\boldsymbol{x}$

This is very classical calculation. The public code $\mathscr{C}_{\text{pub}}$ is alternant, and hence is well–known to have a parity–check matrix defined over $\mathbb{F}_{q^2}$ of the form

$$\boldsymbol{H}_{\text{pub}} = \begin{pmatrix} y_1 & \cdots & y_n \\ x_1 y_1 & \cdots & x_n y_n \\ \vdots & & \vdots \\ x_1^{r-1} y_1 & \cdots & x_n^{r-1} y_n \end{pmatrix}.$$

Denote by $\boldsymbol{G}_{\text{pub}}$ a generator matrix of $\mathscr{C}_{\text{pub}}$. Then, since the x_i's are known, then the y_i's can be computed by solving the linear system

$$\boldsymbol{H}_{\text{pub}} \cdot \boldsymbol{G}_{\text{pub}}^{\top} = 0.$$

4.7.3 Recovering $\boldsymbol{x}, \boldsymbol{y}$ from $\boldsymbol{x}_{\mathcal{I}}, \boldsymbol{y}_{\mathcal{I}}$

After a suitable reordering of the indexes, one can suppose that $\mathcal{I} = \{s, s+1, \ldots, n\}$. Hence, the entries $x_1, \ldots, x_{s-1}$ of $\boldsymbol{x}$ and $y_1, \ldots, y_{s-1}$ are known. Let us explain how to compute x_s, y_s. Set $\mathcal{I}' \stackrel{\text{def}}{=} \mathcal{I} \setminus \{s\}$. Thus, let $\boldsymbol{G}(\mathcal{I}')$ be a generator matrix of $\mathscr{A}_r(\boldsymbol{x}_{\mathcal{I}'}, \boldsymbol{y}_{\mathcal{I}'})$, which is nothing by $\mathcal{S}_{\mathcal{I}'}(\mathscr{C}_{\text{pub}})$. Using the notation of the previous section, we have

$$\begin{pmatrix} y_1 & \cdots & y_s \\ x_1 y_1 & \cdots & x_s y_s \\ \vdots & & \vdots \\ x_1^{r-1} y_1 & \cdots & x_s^{r-1} y_s \end{pmatrix} \cdot \boldsymbol{G}(\mathcal{I}') = 0.$$

In the above identity, all the $x_i's$ and $y_i's$ are known but x_s, y_s. The entry y_s can be found by solving the linear system

$$\begin{pmatrix} y_1 & \cdots & y_s \end{pmatrix} \cdot \boldsymbol{G}(\mathcal{I}') = 0.$$

Then, x_s can be deduced by solving the linear system

$$\begin{pmatrix} x_1 y_1 & \cdots & x_s y_s \end{pmatrix} \cdot \boldsymbol{G}(\mathcal{I}') = 0.$$

By this manner, we can iteratively recover the entries $x_{s+1}, \ldots, x_n$ and $y_{s+1}, \ldots, y_n$. The only constraint is that $\mathcal{I}$ should be small enough so that $\mathcal{S}_{\mathcal{I}}(\mathscr{C}_{\text{pub}})$ is nonzero. But this always holds true for the choices of $\mathcal{I}$ we made in the previous sections.

4.8 Comparison with a previous attack

First, let us remind the attack on Wild Goppa codes over quadratic extensions [13]. This attack concerns some subclass of alternant codes called *wild Goppa codes*. For such codes a distinguisher exists which permits to compute a filtration of the public code. Hence, after some computations, we obtain the subcode $\mathscr{A}_{r+q+1}(\boldsymbol{x}, \boldsymbol{y})$ of the public code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$. Then, according to Heuristic 27, the computation of a conductor permits to get the code $\mathscr{NT}(\boldsymbol{x})$. As soon as $\mathscr{NT}(\boldsymbol{x})$ is known, the recovery of the secret is easy. Note that, the use of the techniques of § 4.7 can significantly simplifies the end of the attack of [13] which was rather technical.

We emphasise that, out of the calculation of $\mathscr{NT}(\boldsymbol{x})$ by computing a conductor which appears in our attack so that in [13], the two attacks remain very different. Indeed, the way one gets a subcode whose conductor into the public code provides $\mathscr{NT}(\boldsymbol{x})$ is based in [13] on a distinguisher which does not work for general alternant codes which are not Goppa codes. In addition, in the present attack, the use of the permutation group is crucial, while it was useless in [13].

5 Complexity of the attack

As explained earlier, we have not been able to provide a complexity analysis of the approach based on polynomial system solving. In particular because the Macaulay matrix in degree 2 of the system turned out to have a surprisingly low rank, showing that this bilinear system was far from being generic. Consequently, we limit our analysis to the first approach based on performing a brute force search on the subcode $\mathscr{D}$.

Since we look for approximate work factors, we will discuss an upper bound on the complexity and not only a big O.

5.1 Complexity of calculation of Schur products

A Schur product $\mathscr{A} \star \mathscr{B}$ of two codes $\mathscr{A}, \mathscr{B}$ of length n and respective dimensions k_a, k_b is computed as follows.

1. Take bases $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{k_a}$ and $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{k_b}$ of $\mathscr{A}$ and $\mathscr{B}$ respectively and construct a matrix $\boldsymbol{M}$ whose rows are all the possible products $\boldsymbol{a}_i \star \boldsymbol{b}_j$, for $1 \leqslant i \leqslant k_a$ and $1 \leqslant j \leqslant k_b$. This matrix has $k_a k_b$ rows and n columns.

2. Perform Gaussian elimination to get a reduced echelon form of $\boldsymbol{M}$.

The cost of the computation of a reduced echelon form of a $s \times n$ matrix is $ns \min(n, s)$ operations in the base field. The cost of the computation of the matrix $\boldsymbol{M}$ is the cost of $k_a k_b$ Schur products of vectors, i.e. $nk_a k_b$ operations in the base field. This leads to an overall calculation of the Schur product equal to

$$nk_a k_b + nk_a k_b \min(n, k_a k_b)$$

operations in the base field. When $k_a k_b \geqslant n$, the cost of the Schur product can be reduced using a probabilistic shortcut described in [11]. It consists in computing an $n \times n$ submatrix of $\boldsymbol{M}$ by choosing some random subset of products $\boldsymbol{a}_i \star \boldsymbol{b}_j$. This permits to reduce the cost of computing a generator matrix in row echelon form of $\mathscr{A} \star \mathscr{B}$ to $2n^3$ operations in the base field.

5.2 Cost of a single iteration of the brute force search

Computing the conductor $\mathbf{Cond}(\mathscr{X}, \mathscr{C}_{\mathrm{pub}})$ consists in computing the code $(\mathscr{X} \star {\mathscr{C}_{\mathrm{pub}}}^{\perp})^{\perp}$. Since our attack consists in computing such conductors for various $\mathscr{X}$'s, one can compute a generator matrix of ${\mathscr{C}_{\mathrm{pub}}}^{\perp}$ once for good. Hence, one can suppose a generator matrix for ${\mathscr{C}_{\mathrm{pub}}}^{\perp}$ is known. Then, according to § 5.1, the calculation of a generator matrix of $\mathscr{X} \star {\mathscr{C}_{\mathrm{pub}}}^{\perp}$ costs at most $2n^3$ operations. Next, note that for most of the iterations, there is no need to deduce a generator matrix in reduced echelon form of $(\mathscr{X} \star {\mathscr{C}_{\mathrm{pub}}}^{\perp})^{\perp}$, since it suffices to evaluate the dimension of $\mathscr{X} \star {\mathscr{C}_{\mathrm{pub}}}^{\perp}$, which is immediate from the generator matrix in reduced echelon form. If the dimension of the code is not the expected one, namely $n - \dim \mathscr{D} = n - 4$, then we skip to the next iteration.

Hence, the overall cost of a single iteration of the brute force search is bounded above by $2n^3$ operations in $\mathbb{F}_q$.

5.3 Complexity of finding $\mathscr{NT}(\boldsymbol{x})$

According to § 4.5, the average number of iterations of the brute force search is $q^{2\mathrm{Codim}\mathscr{D}}$, that is $q^{\frac{4q}{|\mathcal{G}|}}$. Thus, we get an overall cost of the first step bounded above by

$$2n^3 q^{\frac{4q}{|\mathcal{G}|}} \text{ operations in } \mathbb{F}_q.$$

Since, $n = \Theta(q^2)$, we get a complexity in $O(n^{3+\frac{2q}{|\mathcal{G}|}})$ operations in $\mathbb{F}_q$ for the computation of $\mathscr{NT}(\boldsymbol{x})$.

5.4 Complexity of deducing $\boldsymbol{x}, \boldsymbol{y}$ from $\mathscr{NT}(\boldsymbol{x})$

The final part of the attack is negligible compared to the previous step. Indeed,

- the computation of $\mathscr{NT}(\boldsymbol{x})^{\star 2}$ costs $O(n^2)$ operations in $\mathbb{F}_q$ (because of Remark 9, one can perform these computations over $\mathbb{F}_q$) since the code has dimension 4;
- the computation of $\mathscr{NT}(\boldsymbol{x})^{\star 2} \cap \mathscr{NT}(\boldsymbol{x})$ boils down to linear algebra and costs $O(n^3)$ operations in $\mathbb{F}_q$;
- The enumeration of the subset of $\mathscr{NT}(\boldsymbol{x}) \otimes \mathbb{F}_{q^2}$ of elements whose first entry is 0 an second one is 1 and computation of their norm costs $O(q^4 n) = O(n^3)$ operations in $\mathbb{F}_{q^2}$. Indeed the affine subspace of $\mathscr{NT}(\boldsymbol{x}) \otimes \mathbb{F}_{q^2}$ which is enumerated has dimension 2 over $\mathbb{F}_{q^2}$ and hence has q^4 elements, while the computation of the component wise norm of a vector costs $O(n)$ operations assuming that the Frobenius $z \mapsto z^q$ can be computed in constant time in $\mathbb{F}_{q^2}$.
- The recovery of $\boldsymbol{y}$ from $\boldsymbol{x}$ boils down to linear algebra and hence can also be done in $O(n^3)$ operations in $\mathbb{F}_{q^2}$. If we have to recover $\boldsymbol{x}, \boldsymbol{y}$ from $\boldsymbol{x}_{\mathcal{I}}, \boldsymbol{y}_{\mathcal{I}}$, it can be done iteratively by solving a system of a constant number of equations, hence the cost of one iteration is bounded by a $O(n^2)$ operations in $\mathbb{F}_{q^2}$, thus the overall cost remains bounded above by a $O(n^3)$ operations in $\mathbb{F}_{q^2}$.

As a conclusion, the second part of the attack is negligible compared to the first one. Hence, we have an approximate work factor of the form

$$2n^3 q^{\frac{4q}{|G|}} \text{operations in } \mathbb{F}_q. \tag{9}$$

5.5 Approximate work factors of the first variant of the attack on DAGS parameters

We make the approximation that operations in $\mathbb{F}_q$ can be done in constant time. Indeed, the base fields of the public keys of DAGS proposal are $\mathbb{F}_{32}$ and $\mathbb{F}_{64}$. For such a field, it is reasonable to store a multiplication and inversion table.

Therefore, we list in Table 3 some approximate work factors for DAGS according to (9). The second column recalls the security levels claimed in [3] for the best possible attack. The last column gives the approximate work factors for the first variant of our attack.

Name	Claimed security level	Work factor of our attack
DAGS_1	128 bits	$\approx 2^{70}$
DAGS_3	192 bits	$\approx 2^{80}$
DAGS_5	256 bits	$\approx 2^{58}$

Table 3: Work factors of the first variant of the attack

6 Implementation

Since the first variant of the attack had too important costs to be tested on our machines, we made two different kind of tests. All the tests have been done using Magma [8] on an Intel® Xeon 2.27 GHz.

1. We tested the first variant of the attack on the toy parameters `DAGS_0`, in order to have evidences that it works. We performed 20 tests, which succeeded in an average time of 2 hours.

2. We tested the second variant based on solving a bilinear system on `DAGS_1`, `_3` and `_5`. We have not been able to break `DAGS_3` keys using this variant of the attack, on the other hand about 100 tests have been performed for `DAGS_1` and `DAGS_5`. The average running time of the attack for `DAGS_1` keys is about 19 minutes and for `DAGS_5` keys is about 35 seconds.

Name	**Claimed security level**	**Average time**
`DAGS_1`	128 bits	19 mn
`DAGS_5`	256 bits	$<$ 1 mn

Table 4: Average times for the second variant of the attack.

Acknowledgements

The authors are supported by French *Agence nationale de la recherche* grants ANR-15-CE39-0013-01 *Manta* and ANR-17-CE39-0007 *CBCrypt*. Élise Barelli is partially supported by a DGA-MRIS scholarship. Computer aided calculations have been performed using software Magma. The authors express their deep gratitude to Jean-Pierre Tillich and Julien Lavauzelle for very helpful comments.

References

[1] M. Baldi, M. Bianchi, and F. Chiaraluce. Security and complexity of the McEliece cryptosystem based on QC-LDPC codes. *IET Information Security*, 7(3):212–220, Sept. 2013.

[2] M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, and D. Schipani. Enhanced public key security for the McEliece cryptosystem. *J. Cryptology*, 29(1):1–27, 2016.

[3] G. Banegas, P. S. Barreto, B. O. Boidje, P.-L. Cayrel, G. N. Dione, K. Gaj, C. T. Gueye, R. Haeussler, J. B. Klamti, O. N'diaye, D. T. Nguyen, E. Persichetti, and J. E. Ricardini. DAGS : Key encapsulation for dyadic GS codes. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/DAGS.zip, Nov. 2017. First round submission to the NIST post-quantum cryptography call.

[4] E. Barelli. On the security of some compact keys for McEliece scheme. In *WCC Workshop on Coding and Cryptography*, Sept. 2017.

[5] T. P. Berger, P.-L. Cayrel, P. Gaborit, and A. Otmani. Reducing key length of the McEliece cryptosystem. In B. Preneel, editor, *Progress in Cryptology - AFRICACRYPT 2009*, volume 5580 of *LNCS*, 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. *Des. Codes Cryptogr.*, 35(1):63–79, 2005.

[7] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Inform. Theory*, 24(3):384–386, May 1978.

[8] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system I: The user language. *J. Symbolic Comput.*, 24(3/4):235–265, 1997.

[9] I. Cascudo, R. Cramer, D. Mirandola, and G. Zémor. Squares of random linear codes. *IEEE Trans. Inform. Theory*, 61(3):1159–1173, March 2015.

[10] A. Couvreur, P. Gaborit, V. Gauthier-Umaña, A. Otmani, and J.-P. Tillich. Distinguisher-based attacks on public-key cryptosystems using Reed-Solomon codes. *Des. Codes Cryptogr.*, 73(2):641–666, 2014.

[11] A. Couvreur, I. Márquez-Corbella, and R. Pellikaan. Cryptanalysis of McEliece cryptosystem based on algebraic geometry codes and their subcodes. *IEEE Trans. Inform. Theory*, 63(8):5404–5418, Aug 2017.

[12] A. Couvreur, A. Otmani, and J.-P. Tillich. Polynomial time attack on wild McEliece over quadratic extensions. In P. Q. Nguyen and E. Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 17–39. Springer Berlin Heidelberg, 2014.

[13] A. Couvreur, A. Otmani, and J.-P. Tillich. Polynomial time attack on wild McEliece over quadratic extensions. *IEEE Trans. Inform. Theory*, 63(1):404–427, Jan 2017.

[14] A. Couvreur, A. Otmani, J.-P. Tillich, and V. Gauthier-Umaña. A polynomial-time attack on the BBCRS scheme. In J. Katz, editor, *Public-Key Cryptography - PKC 2015*, volume 9020 of *LNCS*, pages 175–193. Springer, 2015.

[15] F. U. de Portzamparc. *Algebraic and Physical Security in Code-Based Cryptography. (Sécurités algébrique et physique en cryptographie fondée sur les codes correcteurs d'erreurs).* PhD thesis, Pierre and Marie Curie University, Paris, France, 2015.

[16] P. Delsarte. On subfield subcodes of modified Reed-Solomon codes. *IEEE Trans. Inform. Theory*, 21(5):575–576, 1975.

[17] J.-C. Faugère, A. Otmani, L. Perret, F. de Portzamparc, and J.-P. Tillich. Folding alternant and Goppa Codes with non-trivial automorphism groups. *IEEE Trans. Inform. Theory*, 62(1):184–198, 2016.

[18] J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 279–298, 2010.

[19] J.-C. Faugère, L. Perret, and F. de Portzamparc. Algebraic attack against variants of McEliece with Goppa polynomial of a special form. In *Advances in Cryptology - ASIACRYPT 2014*, volume 8873 of *LNCS*, pages 21–41, Kaoshiung, Taiwan, R.O.C., Dec. 2014. Springer.

[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, Mar. 2005.

[21] D. Goss. *Basic Structures of Function Field arithmetic*, volume 35 of *Ergebnisse der Mathematik und ihrer Grenzgebiete (3) [Results in Mathematics and Related Areas (3)]*. Springer-Verlag, Berlin, 1996.

[22] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North–Holland, Amsterdam, fifth edition, 1986.

[23] R. J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.

[24] R. Misoczki and P. Barreto. Compact McEliece keys from Goppa codes. In *Selected Areas in Cryptography*, Calgary, Canada, Aug. 13-14 2009.

[25] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. L. M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 2069–2073, 2013.

[26] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.

[27] 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, Apr. 2008. LMIB Beihang University.

[28] E. Persichetti. Compact McEliece keys based on quasi-dyadic Srivastava codes. *J. Math. Cryptol.*, 6(2):149–169, 2012.

[29] V. M. Sidelnikov and S. Shestakov. On the insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discrete Math. Appl.*, 1(4):439–444, 1992.

[30] Y. Wang. Quantum resistant random linear code based public key encryption scheme RLCE. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2016*, pages 2519–2523, Barcelona, Spain, July 2016. IEEE.

[31] C. Wieschebrink. Two NP-complete problems in coding theory with an application in code based cryptography. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 1733–1737, 2006.

[32] C. Wieschebrink. Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes. In *Post-Quantum Cryptography 2010*, volume 6061 of *LNCS*, pages 61–72. Springer, 2010.