# DAGS: Key encapsulation using dyadic GS codes

Gustavo Banegas, Paulo S. l. m. Barreto, Brice Odilon Boidje, Pierre-Louis Cayrel, Gilbert Ndollane Dione, Kris Gaj, Cheikh Thiecoumba Gueye, Richard Haeussler, Jean Belo Klamti, Ousmane N'diaye, et al.

**Research Article**

Gustavo Banegas, Paulo S. L. M. Barreto, Brice Odilon Boidje, Pierre-Louis Cayrel, Gilbert Ndollane Dione, Kris Gaj, Cheikh Thiécoumba Gueye, Richard Haeussler, Jean Belo Klamti, Ousmane N'diaye, Duc Tri Nguyen, Edoardo Persichetti* and Jefferson E. Ricardini

# DAGS: Key encapsulation using dyadic GS codes

https://doi.org/10.1515/jmc-2018-0027
Received February 23, 2018; accepted August 15, 2018

**Abstract:** Code-based cryptography is one of the main areas of interest for NIST's Post-Quantum Cryptography Standardization call. In this paper, we introduce DAGS, a Key Encapsulation Mechanism (KEM) based on quasi-dyadic generalized Srivastava codes. The scheme is proved to be IND-CCA secure in both random oracle model and quantum random oracle model. We believe that DAGS will offer competitive performance, especially when compared with other existing code-based schemes, and represent a valid candidate for post-quantum standardization.

**Keywords:** Post-quantum cryptography, code-based cryptography, key exchange

**MSC 2010:** 94B05, 11T71, 14G50, 94A60

**Communicated by:** Tran van Trung

## 1 Introduction

The availability of large-scale quantum computers is getting ever closer to reality, and with it, all of the public-key cryptosystems currently in use, which rely on number theory problems (e.g., factorization) and discrete logarithm problems will become obsolete [41]. Therefore, it is of extreme importance to be able to offer a credible alternative that can resist attackers equipped with quantum technology. In this regard, NIST's call for proposals for post-quantum standardization is a further reassurance about the need for solid post-quantum proposals. Furthermore, considering the desired life of the encrypted data, and the lengthy timeframe for such

**Gustavo Banegas,** Technische Universiteit Eindhoven, Eindhoven, Netherlands, e-mail: gustavo@cryptme.in
**Paulo S. L. M. Barreto,** University of Washington Tacoma, Tacoma, USA, e-mail: pbarreto@gmail.com
**Brice Odilon Boidje, Gilbert Ndollane Dione, Cheikh Thiécoumba Gueye, Jean Belo Klamti, Ousmane N'diaye,** Laboratoire d'Algebre, de Cryptographie, de Géométrie Algébrique et Applications, Université Cheikh Anta Diop, Dakar, Senegal, e-mail: boidjebo@gmail.com, dionegilbert@gmail.com, cheikht.gueye@ucad.edu.sn, jklamty@gmail.com, ouzdeville@gmail.com
**Pierre-Louis Cayrel,** Laboratoire Hubert Curien, Université Jean Monnet, Saint-Etienne, France, e-mail: pierre.louis.cayrel@univ-st-etienne.fr
**Kris Gaj, Richard Haeussler, Duc Tri Nguyen,** George Mason University, Washington D. C., USA, e-mail: kgaj@gmu.edu, rhaeussl@masonlive.gmu.edu, cothannguyen@gmail.com
***Corresponding author: Edoardo Persichetti,** Department of Mathematical Sciences, Florida Atlantic University, Boca Raton, USA, e-mail: epersichetti@fau.edu
**Jefferson E. Ricardini,** Universidade de São Paulo, São Paulo, Brazil, e-mail: jeffricardini@gmail.com

a complex standardization process, it is clear how convincing research work in post-quantum cryptography is not only necessary, but also urgent.

Code-based cryptography is one of the main candidates for this task. The area is generally based on the syndrome decoding problem [10], which has shown no vulnerabilities to quantum attacks over the years. Since McEliece's seminal work [30] in 1978, many variants and modifications have been proposed, trying to balance security and efficiency and in particular dealing with inherent flaws such as the large size of the public keys. In fact, while the original McEliece's cryptosystem (based on binary Goppa codes) is still formally unbroken, it features a key of several tenths of kilobytes, which has effectively prevented its use in many applications.

There are currently two main trends to deal with this issue, and they both involve structured matrices: the first, is based on "traditional" algebraic codes, and in particular alternant codes such as Goppa or generalized Srivastava codes; the second suggests to use sparse matrices as in LDPC/MDPC codes [3, 32]. This work builds on the former approach, initiated in 2009 by Berger et al. [9], who proposed Quasi-Cyclic (QC) codes, and Misoczki and Barreto [31], suggesting Quasi-Dyadic (QD) codes instead (later generalized to Quasi-Monoidic (QM) codes [8]). Both proposals feature very compact public keys due to the introduction of the extra algebraic structure, but unfortunately this also leads to a vulnerability. Indeed, Faugère et al. [22] devised a clever attack (known simply as FOPT) which exploits the algebraic structure to build a system of equations, which can successively be solved using Gröbner bases techniques. As a result, the QC proposal is heavily compromised, while the QD/QM approach needs to be treated with caution. In fact, for a proper choice of parameters, it is still possible to design secure schemes, using for instance binary Goppa codes, or Generalized Srivastava (GS) codes as suggested by Persichetti in [36].

**Our contribution.** In this paper, we present DAGS[1], a key encapsulation mechanism that follows the QD approach using GS codes. KEMs are the primitive favored by NIST for key exchange schemes, and can be used to build encryption schemes, for example using the hybrid encryption paradigm introduced by Cramer and Shoup [18]. To the best of our knowledge, this is the first code-based KEM that uses quasi-dyadic codes. Another NIST submission, named BIG QUAKE [44], proposes a scheme based on quasi-cyclic codes.

Our KEM achieves IND-CCA security, following the recent framework by Kiltz et al. [27], and features compact public keys and efficient encapsulation and decapsulation algorithms. We modulate our parameters to achieve an efficient scheme, while at the same time keeping out of range of the FOPT attack. We provide an initial performance analysis of our scheme as well as access to our reference code; the team is currently working at several additional, optimized implementations, using C++, assembly language, and hardware (FPGA).

**Related work.** We show that our proposal compares well with other post-quantum KEMs. These include the classic McEliece approach [47], as well as more recent proposals such as BIKE [45] and the aforementioned BIG QUAKE.

The "Classic McEliece" project is an evolution of the well-known McBits [12] (based on the work of Persichetti [37]), and benefits from a well-understood security assessment but suffers from the usual public key size issue. BIKE, a protocol based on QC-MDPC codes, is the result of a merge between two independently published works with similar background, namely CAKE [7] and Ouroboros [19]. The scheme possesses some

1 DAGS is not only an acronym but also one of the names for the Elder Futhark rune pictured above. The shape of the rune recalls the dyadic property of the matrices at the core of our scheme.

very nice features like compact keys and an easy implementation approach, but has currently some potential drawbacks. In fact, the QC-MDPC encryption scheme on which it is based is susceptible to a reaction attack by Guo, Johansson and Stankovski (GJS) [25], and thus the protocol is forced to employ ephemeral keys. Moreover, due to its non-trivial Decoding Failure Rate (DFR), achieving IND-CCA security becomes very hard, so that the BIKE protocol only claims to be IND-CPA secure.

Finally, BIG QUAKE continues the line of work of [9] and proposes to use quasi-cyclic Goppa codes. Due to the particular nature of the FOPT attack and its successors [21], it seems harder to provide security with this approach, and the protocol chooses very large parameters in order to do so. We will discuss attack and parameters in Section 5.

More distantly related are lattice-based schemes like NewHope [2] and Frodo [14], based respectively on LWE and its ring variant. While these schemes are not necessarily a direct comparison term, it is nice to observe that DAGS offers comparable performance.

**Organization of the paper.** The paper is organized as follows. We start by giving some preliminary notions in Section 2. We describe the DAGS protocol in Section 3, and we discuss its provable security in Section 4, showing that DAGS is IND-CCA secure in the random oracle model as well as the quantum random oracle model. Section 5 features a discussion about practical security and known attacks, which include general decoding attacks (information set decoding and the like) as well as algebraic attacks; we then present parameters for the scheme. Performance details are given in Section 6. Finally, we conclude in Section 7.

# 2 Preliminaries

## 2.1 Notation

We will use the following conventions throughout the rest of the paper:

| | |
|---|---|
| $a$ | a constant |
| $\boldsymbol{a}$ | a vector |
| $A$ | a matrix |
| $\mathcal{A}$ | an algorithm or (hash) function |
| $\mathsf{A}$ | a set |
| $(\boldsymbol{a} \parallel \boldsymbol{b})$ | the concatenation of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ |
| $\mathrm{Diag}(\boldsymbol{a})$ | the diagonal matrix formed by the vector $\boldsymbol{a}$ |
| $I_n$ | the $n \times n$ identity matrix |
| $\xleftarrow{\$}$ | choosing a random element from a set or distribution |
| $\ell$ | the length of a shared symmetric key |

## 2.2 Linear codes

We briefly recall some fundamental notions from coding theory. The *Hamming weight* of a vector $\boldsymbol{x} \in \mathbb{F}_q^n$ is given by the number $\mathrm{wt}(\boldsymbol{x})$ of its non-zero components. We define a linear code using the metric induced by the Hamming weight.

**Definition 2.1.** An $[n, k]$-*linear code* $\mathsf{C}$ of length $n$ and dimension $k$ over $\mathbb{F}_q$ is a $k$-dimensional vector subspace of $\mathbb{F}_q^n$.

A linear code can be represented by means of a matrix $G \in \mathbb{F}_q^{k \times n}$, called *generator matrix*, whose rows form a basis for the vector space defining the code. Alternatively, a linear code can also be represented as kernel of a matrix $H \in \mathbb{F}_q^{(n-k) \times n}$, known as *parity-check matrix*, i.e. $\mathsf{C} = \{\boldsymbol{c} : H\boldsymbol{c}^T = 0\}$. Thanks to the generator matrix, we can easily define the codeword corresponding to a vector $\boldsymbol{\mu} \in \mathbb{F}_q^k$ as $\boldsymbol{\mu}G$. Finally, we call *syndrome* of a vector $\boldsymbol{c} \in \mathbb{F}_q^n$ the vector $H\boldsymbol{c}^T$.

### 2.3 Structured matrices and GS codes

**Definition 2.2.** Given a ring R (in our case the finite field $\mathbb{F}_{q^m}$) and a vector $\boldsymbol{h} = (h_0, \ldots, h_{n-1}) \in \mathsf{R}^n$, the *dyadic* matrix $\Delta(\boldsymbol{h}) \in \mathsf{R}^{n\times n}$ is the symmetric matrix with components $\Delta_{ij} = h_{i\oplus j}$, where $\oplus$ stands for bitwise exclusive-or on the binary representations of the indices. The sequence $\boldsymbol{h}$ is called its *signature*. Moreover, $\Delta(r, \boldsymbol{h})$ denotes the matrix $\Delta(\boldsymbol{h})$ truncated to its first $r$ rows. Finally, we call a matrix *quasi-dyadic* if it is a block matrix whose component blocks are $r \times r$ dyadic submatrices.

If $n$ is a power of 2, then every $2^l \times 2^l$ dyadic matrix can be described recursively as

$$M = \begin{pmatrix} A & B \\ B & A \end{pmatrix},$$

where each block is a $2^{l-1} \times 2^{l-1}$ dyadic matrix. Note that by definition any $1 \times 1$ matrix is trivially dyadic.

**Definition 2.3.** For $m, n, s, t \in \mathbb{N}$ and a prime power $q$, let $\alpha_1, \ldots, \alpha_n$ and $w_1, \ldots, w_s$ be $n + s$ distinct elements of $\mathbb{F}_{q^m}$, and $z_1, \ldots, z_n$ be non-zero elements of $\mathbb{F}_{q^m}$. The *generalized Srivastava code* of order $st$ and length $n$ is defined by a parity-check matrix of the form

$$H = \begin{pmatrix} H_1 \\ H_2 \\ \vdots \\ H_s \end{pmatrix},$$

where each block is given by

$$H_i = \begin{pmatrix} \frac{z_1}{\alpha_1 - w_i} & \cdots & \frac{z_n}{\alpha_n - w_i} \\ \frac{z_1}{(\alpha_1 - w_i)^2} & \cdots & \frac{z_n}{(\alpha_n - w_i)^2} \\ \vdots & \ddots & \vdots \\ \frac{z_1}{(\alpha_1 - w_i)^t} & \cdots & \frac{z_n}{(\alpha_n - w_i)^t} \end{pmatrix}.$$

The parameters for such a code are the length $n \le q^m - s$, dimension $k \ge n - mst$ and minimum distance $d \ge st + 1$. GS codes are part of the family of alternant codes, and therefore benefit of an efficient decoding algorithm; according to Sarwate [40, Corollary 2] the complexity of decoding is $\mathcal{O}(n \log^2 n)$, which is the same as for Goppa codes. Moreover, it can be easily proved that every GS code with $t = 1$ is a Goppa code. More information about this class of codes can be found in [29, Chapter 12, Section 6].

## 3 Construction

The core idea of DAGS is to use GS codes which are defined by matrices in quasi-dyadic form. In particular, the public key of the scheme is the generator matrix of such a code, which, being quasi-dyadic, can be described using just the signature of each block. This allows to obtain a very compact public key. Now, it can be easily proved that every GS code with $t = 1$ is a Goppa code, and we know [29, Chapter 12, Proposition 5] that Goppa codes admit a parity-check matrix in Cauchy form under certain conditions (the generator polynomial has to be monic and without multiple zeros). By Cauchy we mean a matrix $C(\boldsymbol{u}, \boldsymbol{v})$ with components $C_{ij} = \frac{1}{u_i - v_j}$.

Misoczki and Barreto showed in [31, Theorem 2] that the intersection of the set of Cauchy matrices with the set of dyadic matrices is not empty if the code is defined over a field of characteristic 2, and the dyadic signature $\boldsymbol{h} = (h_0, \ldots, h_{n-1})$ satisfies the following "fundamental" equation:

$$\frac{1}{h_{i\oplus j}} = \frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0}. \tag{3.1}$$

On the other hand, it is evident from Definition 2.3 that if we permute the rows of $H$ to constitute $s \times n$ blocks of the form

$$H_i = \begin{pmatrix} \frac{z_1}{(\alpha_1 - w_1)^i} & \cdots & \frac{z_n}{(\alpha_n - w_1)^i} \\ \frac{z_1}{(\alpha_1 - w_2)^i} & \cdots & \frac{z_n}{(\alpha_n - w_2)^i} \\ \vdots & \ddots & \vdots \\ \frac{z_1}{(\alpha_1 - w_s)^i} & \cdots & \frac{z_n}{(\alpha_n - w_s)^i} \end{pmatrix},$$

we obtain an equivalent parity-check matrix for a GS code, given by

$$\hat{H} = \begin{pmatrix} \hat{H}_1 \\ \hat{H}_2 \\ \vdots \\ \hat{H}_t \end{pmatrix}.$$

The key generation process exploits first of all the fundamental equation to build a Cauchy matrix. The matrix is then successively powered (element by element) forming several blocks which are superimposed and then multiplied by a random diagonal matrix. Thanks to the observation above, we have now formed the matrix $\hat{H}$, where for ease of notation we use $\boldsymbol{u}$ and $\boldsymbol{v}$ to denote the vectors of elements $w_1, \ldots, w_s$ and $\alpha_1, \ldots, \alpha_n$, respectively. Finally, the resulting matrix is projected onto the base field (as usual for alternant codes) and row-reduced to systematic form to form the public key. The process will be described in detail in the next section: note that this is essentially the same as in [36], to which we refer the readers looking for additional details about dyadic GS codes and the key generation process.

## 3.1 Algorithms

We are now ready to introduce the three algorithms that form DAGS. System parameters are the code length $n$ and dimension $k$, the values $s$ and $t$ which define a GS code, the cardinality of the base field $q$ and the degree of the field extension $m$. In addition, we have $k = k' + k''$, where $k'$ is arbitrary and is set to be "small". In practice, the value of $k'$ depends on the base field and is such that a vector of length $k'$ provides at least 256 bits of entropy. This also makes the hash functions (see below) easy to compute, and ensures that the overhead due to the IND-CCA2 security in the QROM is minimal.

DAGS is a key encapsulation mechanism and as such it is composed of three algorithms – Key Generation, Encapsulation and Decapsulation – which will present below in the respective order.

**Algorithm 1** (Key Generation).

(1) Generate the dyadic signature $\boldsymbol{h}$:
   (a) Choose a random non-zero distinct $h_0$ and $h_j$ for $j = 2^l$, $l = 0, \ldots, \lfloor \log q^m \rfloor$.
   (b) Form the remaining elements using (3.1).
   (c) Return a selection[2] of blocks of dimension $s$ up to length $n$.

(2) Build the Cauchy support:
   (a) Choose a random[3] offset $\omega \xleftarrow{\$} \mathbb{F}_{q^m}$.
   (b) Compute $u_i = \frac{1}{h_i} + \omega$ for $i = 0, \ldots, s-1$.
   (c) Compute $v_j = \frac{1}{h_j} + \frac{1}{h_0} + \omega$ for $j = 0, \ldots, n-1$.
   (d) Set $\boldsymbol{u} = (u_0, \ldots, u_{s-1})$ and $\boldsymbol{v} = (v_0, \ldots, v_{n-1})$.

(3) Form the Cauchy matrix $\hat{H}_1 = C(\boldsymbol{u}, \boldsymbol{v})$.

(4) Build $\hat{H}_i$, $i = 2, \ldots, t$, by raising each element of $\hat{H}_1$ to the power of $i$.

**2** Making sure to exclude any block containing an undefined entry.

**3** See Appendix A for restrictions about the choice of the offset.

(5) Superimpose the blocks $\hat{H}_i$ in ascending order to form matrix $\hat{H}$.
(6) Generate the vector $\boldsymbol{z}$ by sampling uniformly at random elements in $\mathbb{F}_{q^m}$ with the restriction $z_{is+j} = z_{is}$ for $i = 0, \ldots, n_0 - 1, j = 0, \ldots, s - 1$.
(7) Set

$$y_j = \frac{z_j}{\prod_{i=0}^{s-1}(u_i - v_j)^t} \quad \text{for } j = 0, \ldots, n-1 \text{ and } \boldsymbol{y} = (y_0, \ldots, y_{n-1}).$$

(8) Form $H = \hat{H} \cdot \mathrm{Diag}(\boldsymbol{z})$.
(9) Project $H$ onto $\mathbb{F}_q$ using the co-trace function; call this $H_{\text{base}}$.
(10) Write $H_{\text{base}}$ in systematic form $(M \mid I_{n-k})$.
(11) The public key is the generator matrix $G = (I_k \mid M^T)$.
(12) The private key is the pair $(\boldsymbol{v}, \boldsymbol{y})$.

The encapsulation and decapsulation algorithms follow the paradigm of [27] to obtain an IND-CCA secure KEM from a PKE, and as such, they make use of two functions $\mathcal{G}\colon \mathbb{F}_q^{k'} \to \mathbb{F}_q^k$ and $\mathcal{H}\colon \mathbb{F}_q^{k'} \to \mathbb{F}_q^{k'}$, respectively an expansion and a compression function, the former with the task of generating randomness for the scheme, the latter to provide "plaintext confirmation". The shared symmetric key is obtained via a third function $\mathcal{K}\colon \{0,1\}^* \to \{0,1\}^\ell$. For more details about randomness generation and how the functions are implemented in practice, see Section 6.2.

**Algorithm 2** (Encapsulation).
(1) Choose $\boldsymbol{m} \xleftarrow{\$} \mathbb{F}_q^{k'}$.
(2) Compute $\boldsymbol{r} = \mathcal{G}(\boldsymbol{m})$ and $\boldsymbol{d} = \mathcal{H}(\boldsymbol{m})$.
(3) Parse $\boldsymbol{r}$ as $(\boldsymbol{\rho} \parallel \boldsymbol{\sigma})$ then set $\boldsymbol{\mu} = (\boldsymbol{\rho} \parallel \boldsymbol{m})$.
(4) Generate the error vector $\boldsymbol{e}$ of length $n$ and weight $w$ from $\boldsymbol{\sigma}$.
(5) Compute $\boldsymbol{c} = \boldsymbol{\mu}G + \boldsymbol{e}$.
(6) Compute $\boldsymbol{k} = \mathcal{K}(\boldsymbol{m})$.
(7) Output the ciphertext $(\boldsymbol{c}, \boldsymbol{d})$; the encapsulated key is $\boldsymbol{k}$.

The decapsulation algorithm consists mainly of decoding the noisy codeword received as part of the ciphertext. This is done using the alternant decoding algorithm described in [29, Chapter 12, Section 9] and requires the parity-check matrix to be in alternant form.

**Algorithm 3** (Decapsulation).
(1) Get the parity-check matrix $H'$ in alternant form from a private key[4].
(2) Use $H'$ to decode $\boldsymbol{c}$ and obtain the codeword $\boldsymbol{\mu}'G$ and the error $\boldsymbol{e}'$.
(3) Output $\perp$ if decoding fails or $\mathrm{wt}(\boldsymbol{e}') \neq w$.
(4) Recover $\boldsymbol{\mu}'$ and parse it as $(\boldsymbol{\rho}' \parallel \boldsymbol{m}')$.
(5) Compute $\boldsymbol{r}' = \mathcal{G}(\boldsymbol{m}')$ and $\boldsymbol{d}' = \mathcal{H}(\boldsymbol{m}')$.
(6) Parse $\boldsymbol{r}'$ as $(\boldsymbol{\rho}'' \parallel \boldsymbol{\sigma}')$.
(7) Generate the error vector $\boldsymbol{e}''$ of length $n$ and weight $w$ from $\boldsymbol{\sigma}'$.
(8) If $\boldsymbol{e}' \neq \boldsymbol{e}'' \vee \boldsymbol{\rho}' \neq \boldsymbol{\rho}'' \vee \boldsymbol{d} \neq \boldsymbol{d}'$, output $\perp$.
(9) Else, compute $\boldsymbol{k} = \mathcal{K}(\boldsymbol{m}')$.
(10) The decapsulated key is $\boldsymbol{k}$.

DAGS is built upon the McEliece cryptosystem, with a notable exception. In fact, we incorporate the "randomized" version of McEliece by Nojima et al. [35] into our scheme. This is extremely beneficial for two distinct aspects: first of all, it allows us to use a much shorter vector $\boldsymbol{m}$ to generate the remaining components of the scheme, greatly improving efficiency. Secondly, it allows us to get tighter security bounds. Note that our protocol differs slightly from the paradigm presented in [27], in the fact that we do not perform a full re-encryption

4 See Section 6.3.

in the decapsulation algorithm. Instead, we simply re-generate the randomness and compare it with the one obtained after decoding. This is possible since, unlike a generic PKE, McEliece decryption reveals the randomness used, in our case $\boldsymbol{e}$ (and $\boldsymbol{\rho}$). It is clear that if the re-generated randomness is equal to the retrieved one, the resulting encryption will also be equal. This allows us to further decrease computation time.

The selection of the parameters for the scheme will be discussed in Section 5.4.

# 4 KEM security

In this section, we discuss some aspects of provable security, and in particular we show that DAGS satisfies the notion of IND-CCA security for KEMs, as defined below.

**Definition 4.1.** The adaptive chosen-ciphertext attack game for a KEM proceeds as follows:
(1) Query a key generation oracle to obtain a public key $pk$.
(2) Make a sequence of calls to a decryption oracle, submitting any string $\boldsymbol{c}$ of the proper length. The oracle will respond with Decaps($sk$, $\boldsymbol{c}$).
(3) Query an encryption oracle. The oracle runs Encaps($pk$) to generate a pair $(\tilde{\boldsymbol{k}}, \tilde{\boldsymbol{c}})$, then chooses a random $b \in \{0, 1\}$ and replies with the "challenge" ciphertext $(\boldsymbol{k}^*, \tilde{\boldsymbol{c}})$ where $\boldsymbol{k}^* = \tilde{\boldsymbol{k}}$ if $b = 1$ or $\boldsymbol{k}^*$ is a random string of length $\ell$ otherwise.
(4) Keep performing decryption queries. If the submitted ciphertext is $\boldsymbol{c}^*$, the oracle will return $\perp$.
(5) Output $b^* \in \{0, 1\}$.
The adversary succeeds if $b^* = b$. More precisely, we define the *advantage* of $\mathcal{A}$ against KEM as

$$\mathrm{Adv}_{\mathrm{KEM}}^{\text{IND-CCA}}(\mathcal{A}, \lambda) = \left|\Pr[b^* = b] - \frac{1}{2}\right|.$$

We say that a KEM is secure if the advantage $\mathrm{Adv}_{\mathrm{KEM}}^{\text{IND-CCA}}$ of any polynomial-time adversary $\mathcal{A}$ in the above CCA attack model is negligible.

Before discussing the IND-CCA security of DAGS, we show that the underlying PKE (i.e. randomized McEliece, see [35]) satisfies a simple property. This will allow us to get better security bounds in our reduction.

**Definition 4.2.** Consider a probabilistic PKE with randomness set $\mathsf{R}$. We say that PKE is $\gamma$-*spread* if for a given key pair ($sk$, $pk$), a plaintext $m$ and an element $y$ in the ciphertext domain, we have

$$\Pr[r \xleftarrow{\$} \mathsf{R} \mid y = \mathrm{Enc}_{pk}(m, r)] \leq 2^{-\gamma}$$

for a certain $\gamma \in \mathbb{R}$.

The definition above is presented as in [27], but note that in fact this corresponds to the notion of $\gamma$-*uniformity* given by Fujisaki and Okamoto in [24], except for a change of constants. In other words, a scheme is $\gamma$-spread if it is $2^{-\gamma}$-uniform.

It was proved in [16] that a simple variant of the (classic) McEliece PKE is $\gamma$-uniform for $\gamma = 2^{-k}$, where $k$ is the code dimension as usual (more in general, $\gamma = q^{-k}$ for a cryptosystem defined over $\mathbb{F}_q$). We can extend this result to our scheme as follows.

**Lemma 4.3.** *Randomized McEliece is $\gamma$-uniform for* $\gamma = \dfrac{q^{-k''}}{\binom{n}{w}}$.

*Proof.* Let $\boldsymbol{y}$ be a generic vector of $\mathbb{F}_q^n$. Then either $\boldsymbol{y}$ is a word at distance $w$ from the code, or it is not. If it is not, the probability of $\boldsymbol{y}$ being a valid ciphertext is clearly exactly 0. On the other hand, suppose $\boldsymbol{y}$ is at distance $w$ from the code; then there is only one choice of $\boldsymbol{\rho}$ and one choice of $\boldsymbol{e}$ that satisfy the equation (since $w$ is below the GV bound), i.e. the probability of $\boldsymbol{y}$ being a valid ciphertext is exactly $1/q^{k''} \cdot 1/\binom{n}{w}$, which concludes the proof. □

We are now ready to present the security results.

**Theorem 4.4.** *Let $\mathcal{A}$ be an IND-CCA adversary against DAGS that makes at most $q_{\mathrm{RO}} = q_{\mathcal{G}} + q_{\mathcal{K}}$ total random oracle queries[5] and $q_{\mathrm{D}}$ decryption queries. Then there exists an IND-CPA adversary $\mathcal{B}$ against PKE, running in approximately the same time as $\mathcal{A}$, such that*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\mathrm{KEM}}(\mathcal{A}) \leq q_{\mathrm{RO}} \cdot 2^{-\gamma} + 3 \cdot \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{PKE}}(\mathcal{B}).$$

*Proof.* The thesis is a consequence of the results presented in [27, Section 3.3]. In fact, our scheme follows the $\mathrm{KEM}^{\perp}_{m}$ framework that consists of applying two generic transformations to a public-key encryption scheme. The first step consists of transforming the IND-CPA encryption scheme into an OW-PCVA (i.e. plaintext and validity checking) scheme. Then, the resulting scheme is transformed into a KEM in a "standard" way. Both proofs are obtained via a sequence of games, and the combination of them shows that breaking IND-CCA security of the KEM would lead to break the IND-CPA security of the underlying encryption scheme. Note that randomized McEliece, instantiated with quasi-dyadic GS codes, presents no correctness error (the value $\delta$ in [27]), which greatly simplifies the resulting bound. □

The value $d$ included in the KEM ciphertext does not contribute to the security result above, but it is a crucial factor to provide security in the Quantum Random Oracle Model (QROM). We present this in the next theorem.

**Theorem 4.5.** *Let $\mathcal{A}$ be a quantum IND-CCA adversary against DAGS that makes at most $q_{\mathrm{RO}} = q_{\mathcal{G}} + q_{\mathcal{K}}$ total quantum random oracle queries[6] and $q_{\mathrm{D}}$ (classical) decryption queries. Then there exists an OW-CPA adversary $\mathcal{B}$ against PKE, running in approximately the same time as $\mathcal{A}$, such that*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\mathrm{KEM}}(\mathcal{A}) \leq 8q_{\mathrm{RO}} \cdot \sqrt{q_{\mathrm{RO}} \cdot \sqrt{\mathrm{Adv}^{\text{OW-CPA}}_{\mathrm{PKE}}(\mathcal{B})}}.$$

*Proof.* The thesis is a consequence of the results presented in Section 4.4 of [27]. In fact, our scheme follows the $\mathrm{QKEM}^{\perp}_{m}$ framework that consists of applying two generic transformations to a public-key encryption scheme. The first step transforming the IND-CPA encryption scheme into an OW-PCVA (i.e. plaintext and validity checking) scheme, is the same as in the previous case. Now, the resulting scheme is transformed into a KEM with techniques suitable for the QROM. The combination of the two proofs shows that breaking IND-CCA security of the KEM would lead to break the OW-CPA security of the underlying encryption scheme. Note, therefore, that the IND-CPA security of the underlying PKE has in this case no further effect on the final result, and can be considered instead just a guarantee that the scheme is indeed OW-CPA secure. The bound obtained is a "simplified" and "concrete" version (as presented by the authors) and, in particular, it is easy to notice that it does not depend on the number of queries $q_{\mathcal{H}}$ presented to the random oracle $\mathcal{H}$. The bound is further simplified since, as above, the underlying PKE presents no correctness error. □

# 5 Practical security and parameters

Having proved that DAGS satisfies the notion of IND-CCA security for KEMs, we now move onto a treatment of practical security issues. In particular, we will briefly present the hard problem on which DAGS is based, and then discuss the main attacks on the scheme and related security concerns.

## 5.1 Hard problems from coding theory

Most of the code-based cryptographic constructions are based on the hardness of the following problem, known as the ($q$-ary) *Syndrome Decoding Problem (SDP)*.

5 Respectively, $q_{\mathcal{G}}$ queries to the random oracle $\mathcal{G}$ and $q_{\mathcal{K}}$ queries to the random oracle $\mathcal{K}$.

6 Same as in Theorem 4.4.

**Problem 5.1.** *Given an* $(n-k) \times n$ *full-rank matrix* $H$ *over* $\mathbb{F}_q$*, a vector* $\boldsymbol{s} \in \mathbb{F}_q^{n-k}$*, and a non-negative integer* $w$*, find a vector* $\boldsymbol{e} \in \mathbb{F}_q^n$ *of weight* $w$ *such that* $H\boldsymbol{e}^T = \boldsymbol{s}$*.*

The corresponding decision problem was proved to be NP-complete in 1978 [10], but only for binary codes. In 1994, Barg proved that this result holds for codes over all finite fields ([5], in Russian, and [6, Theorem 4.1]).

In addition, many schemes (including the original McEliece proposal) require the following computational assumption.

**Assumption 1.** *The public matrix output by the key generation algorithm is computationally indistinguishable from a uniformly chosen matrix of the same size.*

The assumption above is historically believed to be true, except for very particular cases. For instance, there exists a distinguisher (Faugère et al. [20]) for cryptographic protocols that make use of high-rate Goppa codes (like the CFS signature scheme [17]). Moreover, it is worth mentioning that the "classical" methods for obtaining an indistinguishable public matrix, such as the use of scrambling matrices $S$ and $P$, are rather outdated and unpractical and can introduce vulnerabilities to the scheme as per the work of Strenzke et al. [42, 43]. Thus, traditionally, the safest method (Biswas and Sendrier, [13]) to obtain the public matrix is simply to compute the systematic form of the private matrix.

## 5.2 Decoding attacks

The main approach for solving SDP is the technique known as Information Set Decoding (ISD), first introduced by Prange [39], which targets directly the error vector and aims at decoding without knowing the underlying structure of the code (i.e. treating the code as truly random). Among several variants and generalizations, Peters showed [38] that it is possible to apply Prange's approach to generic $q$-ary codes. Other approaches such as statistical decoding [1, 33] are usually considered less efficient. Thus, when choosing parameters, we will focus mainly on defeating attacks of the ISD family.

Hamdaoui and Sendrier in [26] provide non-asymptotic complexity estimates for ISD in the binary case. For codes over $\mathbb{F}_q$, instead, a bound is given in [34], which extends the work of Peters. For a practical evaluation of the ISD running times and corresponding security level, we used Peters's ISDFQ script [46].

**Quantum speedup.** Bernstein in [11] shows that Grover's algorithm applies to ISD-like algorithms, effectively halving the asymptotic exponent in the complexity estimates. Later, it was proved in [28] that several variants of ISD have the potential to achieve a better exponent, however the improvement was disappointingly away from the factor of 2 that could be expected. For this reason, we simply treat the best quantum attack on our scheme to be "traditional" ISD (Prange) combined with Grover search.

## 5.3 Algebraic attacks

While, as we discussed above, recovering a private matrix from a public one is in general a very difficult problem, the presence of special algebraic properties and additional structure in the code can have a considerable effect in lowering this difficulty. It turns out that, in the case of alternant codes for instance, there are indeed efficient methods that exploit this issue.

**Solving systems of equations.** A very effective structural attack was introduced by Faugère et al. in [22]. The attack (for convenience referred to as FOPT) relies on the simple property $H \cdot G^T = 0$ to build an algebraic system, using then Gröbner bases techniques to solve it. Note that this applies in principal to every linear code, but the system of equations is in general way too large to be solved in practice. It is then the special properties of alternant codes, as we mentioned above, that make the attack possible, by considerably reducing the number of unknowns of the system.

The attack was originally aimed at two variants of McEliece, introduced respectively in [9] and [31]. The first variant, using quasi-cyclic codes, was easily broken in all proposed parameters and falls out of the scope

| $q$ | $m$ | $n$ | $k$ | $n_0$ | $\ell$ | $n_{X'}$ | $n_{Y'}$ | Time (s) / Operations |
|---|---|---|---|---|---|---|---|---|
| 2 | 16 | 3584 | 1536 | 56 | $2^6$ | 60 | 15 | N/A |
| $2^2$ | 8 | 3584 | 1536 | 56 | $2^6$ | 60 | 7 | 1776.3 / $2^{34.2}$ |
| $2^4$ | 4 | 2048 | 1024 | 32 | $2^6$ | 36 | 3 | 0.5 / $2^{22.1}$ |
| $2^8$ | 2 | 1280 | 768 | 20 | $2^6$ | 24 | 1 | 0.03 / $2^{16.7}$ |
| $2^8$ | 2 | 640 | 512 | 10 | $2^6$ | 14 | 1 | 0.03 / $2^{15.9}$ |
| $2^8$ | 2 | 768 | 512 | 6 | $2^7$ | 11 | 1 | 0.02 / $2^{15.4}$ |
| $2^8$ | 2 | 1024 | 512 | 4 | $2^8$ | 10 | 1 | 0.11 / $2^{19.2}$ |
| $2^8$ | 2 | 512 | 256 | 4 | $2^7$ | 9 | 1 | 0.06 / $2^{17.7}$ |
| $2^8$ | 2 | 640 | 384 | 5 | $2^7$ | 10 | 1 | 0.02 / $2^{14.5}$ |
| $2^8$ | 2 | 768 | 512 | 6 | $2^7$ | 11 | 1 | 0.01 / $2^{16.6}$ |
| $2^8$ | 2 | 1280 | 768 | 5 | $2^8$ | 11 | 1 | 0.05 / $2^{17.5}$ |
| $2^8$ | 2 | 1536 | 1024 | 6 | $2^8$ | 12 | 1 | 0.06 / $2^{17.8}$ |
| $2^4$ | 4 | 4096 | 3584 | 32 | $2^7$ | 37 | 3 | 7.1 / $2^{26.1}$ |
| $2^8$ | 2 | 3072 | 2048 | 6 | $2^9$ | 13 | 1 | 0.15 / $2^{19.7}$ |

**Table 1:** Details of FOPT applied to quasi-dyadic Goppa codes [23].

of this paper. The second variant, instead, only considered quasi-dyadic Goppa codes. In this case too, most of the parameters proposed have been broken, except for the binary case (i.e. base field $\mathbb{F}_2$). This was, in truth, not connected to the base field per se, but rather depended on the fact that, with a smaller base field, the authors provided a much higher extension degree $m$. This is because, probably for comparison reasons, all the proposed parameters were chosen so that the value $q^m = 2^{16}$ was kept constant. As it turns out, the extension degree $m$ plays a key role in evaluating the complexity of the attack.

**Attack complexity.** Following up on their own work, the authors in [23] produced a paper which analyzes the attack in detail, with the aim of evaluating its complexity at least somewhat rigorously. At the core of the attack, there is an affine bilinear system, which is derived from the initial system of equations by applying various algebraic relations due to the quasi-dyadic structure. This bilinear system has $n_{X'} + n_{Y'}$ variables, where these are, respectively, the number of $X$ and $Y$ "free" variables (after applying the relations) of an alternant parity-check matrix $H$ with $H_{ij} = Y_j X_j^i$. Moreover, the *degree of regularity* (i.e. the maximal degree of the polynomials appearing during the computation) is bounded above by $1 + \min\{n_{X'}, n_{Y'}\}$. It is shown that this number dominates computation time, and so it is crucial to correctly evaluate it in our case. In fact, for the original proposal based on Goppa codes [31], we have $n_{X'} = n_0 - 2 + \log_2(\ell)$, where $\ell$ is the dyadic order and $n_0 = n/\ell$ is the number of dyadic blocks, and $n_{Y'} = m - 1$. We report an excerpt of some numbers from the paper in Table 1.

It is possible to observe several facts. In every set of parameters, for instance, $n_{X'} \gg n_{Y'}$, and so $n_{Y'} = m-1$ is the most important number here. In other words, the degree of the extension field is crucial in evaluating the complexity of the attack, as we mentioned above. As a confirmation, it is easy to notice that all parameters were broken very easily when this is extremely small (1 in most cases), while the running time scales accordingly when $m$ grows. In fact, the attack could not be performed in practice on the first set of parameters (hence the N/A).

The first three groups of parameters are taken from the preliminary (unpublished) version of [31, Tables 2, 3 and 5, respectively], while the last group consists of some ad hoc parameters generated by the FOPT authors. It stands out the absence of parameters from [31, Table 4]. In fact, all of these parameters used $\mathbb{F}_2$ as base field and thus could not be broken (at least not without very long computations), just like for the case of the first set. As a result, an updated version of [31] was produced for publication, in which the insecure parameters are removed and only the binary sets (those of [31, Table 4]) appear.

Towards the end of [23], the authors present a bound on the theoretical complexity of computing a Gröbner base of the affine bilinear system which is at the core of the attack. They then evaluate this bound and compare it with the number of operations required in practice (last column of Table 1). The bound is

given by

$$T_{\text{theo}} \approx \Big( \sum_{\substack{d_1+d_2=D,\\ 1\le d_1,d_2\le D-1}} (\dim R_{d_1,d_2} - [t_1^{d_1} t_2^{d_2}] HS(t_1, t_2)) \dim R_{d_1,d_2} \Big), \tag{5.1}$$

where $D$ is the degree of regularity of the system, i.e.

$$\dim R_{d_1,d_2} = \binom{d_1 + n_{X'}}{d_1} \binom{d_2 + n_{X'}}{d_2},$$

and $[t_1^{d_1} t_2^{d_2}] HS(t_1, t_2)$ indicates the coefficient of the term $[t_1^{d_1} t_2^{d_2}]$ in the Hilbert bi-series $HS(t_1, t_2)$, as defined in [23, Appendix A].

As it turns out this bound is quite loose, being sometimes above and sometimes below the experimental results, depending on which set of parameters is considered. As such, it is to be read as a grossly approximate indication of the expected complexity of a parameter set, and it only allows to have a rough idea of the security provided for each set. Nevertheless, since are able to compute the bound for all DAGS proposed parameters, we will keep this number in mind when proposing parameters (Section 5.4), to make sure our choices are at least not obviously insecure.

As a bottom-line, it is clear that the complexity of the attack scales somewhat proportionally to the value $m - 1$ which defines the dimension of the solution space. The FOPT authors point out that any scheme for which this dimension is less or equal to 20 should be within the scope of the attack.

Since GS codes are also alternant codes, the attack can be applied to our proposal as well. There is, however, one very important difference to keep in mind. In fact, it is shown in [36] that, thanks to the particular structure of GS codes, the dimension of the solution space is defined by $mt - 1$, rather than $m - 1$. This provides greater flexibility when designing parameters for the code, and it allows, in particular, to "rest the weight" of the attack on two shoulders rather than just one. Thus we are able to modulate the parameters and keep the extension degree $m$ small while still achieving a large dimension for the solution space. We will discuss parameter selection in detail in Section 5.4 as already mentioned.

**Folded codes.** Recently, an extension of the FOPT attack appeared in [21]. In this work, the authors introduce a new technique called "folding", and show that it is possible to reduce the complexity of the FOPT attack to the complexity of attacking a smaller code (the "folded" code). This is a consequence of the strong properties of the automorphism group that is present in the alternant codes used. The attack turns out to be very efficient against Goppa codes, as it is possible to recover a folded code which is also a Goppa code. As a consequence, it is possible to tweak the attack to solve a different, augmented system of equations (named $\mathrm{G}_{X,Y'}$), rather than the "basic" one which is aimed at generic alternant codes (called $\mathrm{A}_{X,Y'}$). Moreover, this can be further refined in the case of *binary* Goppa codes, leading to a third system of equations referred to as $\mathrm{McE}_{X,Y'}$. In parallel, the authors present a new method called "structural elimination" that manages to eliminate a considerable number of variables, at the price of an increased degree in the equations considered. Solving the "eliminated" systems (called respectively $\mathrm{elimA}_{X',Y'}$, $\mathrm{elimG}_{X',Y'}$ and $\mathrm{elimMcE}_{X',Y'}$) often proves a more efficient choice, but the authors do occasionally use the non-eliminated systems when it is more convenient to do so.

The paper concentrates on attacking several parameters that were proposed for signature schemes and encryption schemes in various follow-up works that build and expand on [9] and [31]. The latter includes, among others, some of the parameters presented in Table 1. It turns out that codes designed to work for signature schemes are very easy to attack (due to their particular nature); however, the situation for encryption is more complex. The authors are able to obtain a speedup in the attack times for previously investigated parameters, but some of the parameters could still not be solved in practice. We report the results in Table 2, where we indicate the type of system chosen to be solved, and we keep some of the previously-shown parameters for ease of comparison.

The authors do not report timings for codes that were already broken with FOPT in negligible time (which is the case for all the parameter sets where $m = 2$). Also, we have decided to exclude from our table parameters that are not relevant to this submission. These include for example the quasi-monoidic codes introduced in [8] (i.e. codes defined over a field $\mathbb{F}_q$ for $q$ not a power of 2).

| $q$ | $m$ | $n$ | $k$ | $n_0$ | $\ell$ | System | New attack | FOPT |
|---|---|---|---|---|---|---|---|---|
| $2^4$ | 4 | 2048 | 1024 | 32 | $2^6$ | elim$A_{X',Y'}$ | 0.01 s | 0.5 s |
| $2^4$ | 4 | 4096 | 3584 | 32 | $2^7$ | elim$A_{X',Y'}$ | 0.01 s | 7.1 s |
| $2^2$ | 8 | 3584 | 1536 | 56 | $2^6$ | elim$A_{X',Y'}$ | 0.04 s | 1776.3 s |
| 2 | 16 | 4864 | 4352 | 152 | $2^5$ | elimMcE$_{X',Y'}$ | 18 s | N/A |
| 2 | 12 | 3200 | 1664 | 25 | $2^7$ | elimMcE$_{X',Y'}$ | $\leq 2^{83.5}$ op | N/A |
| 2 | 14 | 5376 | 3584 | 42 | $2^7$ | elimMcE$_{X',Y'}$ | $\leq 2^{96.1}$ op | N/A |
| 2 | 15 | 11264 | 3584 | 22 | $2^9$ | elimMcE$_{X',Y'}$ | $\leq 2^{146}$ op | N/A |
| 2 | 16 | 6912 | 2816 | 27 | $2^8$ | elimMcE$_{X',Y'}$ | $\leq 2^{168}$ op | N/A |
| 2 | 16 | 8192 | 4096 | 32 | $2^8$ | elimMcE$_{X',Y'}$ | $\leq 2^{157}$ op | N/A |

**Table 2:** Details of folding attack applied to quasi-dyadic Goppa codes [23].

This table confirms our intuition that high values of $m$ result in a high number of operations, and that complexity increases somewhat proportionally to this value. Note that the last five sets of parameters were not broken in practice and the estimated complexity is always quite high. It is not clear what the authors mean by $\leq$, but it is reasonable to assume that the actual complexity would not be dramatically smaller than the indicated value, and thus at least $2^{80}$ in all cases. Consequently, the claim that parameters with $m - 1 \leq 20$ are "within the scope of the attack" looks now perhaps a bit optimistic.

The fourth set of parameters seem to contradict our intuition, since it was broken in practice with relative ease even though $m = 16$. However, it is possible to see that this is a code with a ridiculously high rate ($k/n$ is very close to1) and in particular, the very large number of blocks $n_0$ clearly stands out. We remark that this set of parameters was chosen ad hoc by the attack authors and in practice such a poor choice of parameters would never be considered. Nevertheless, it gives us the confirmation (if needed) that high-rate codes are a bad choice not only for ISD-like attacks, but for structural attacks also.

The authors did not present any explicit result against GS codes and, in particular, it is not known whether a folded GS code is still a GS code. Thus, the attack in this case is limited to solving the generic system $A_{X,Y'}$ (or elim$A_{X',Y'}$) and does not benefit from the speedups which are specific to (binary) Goppa codes. For these reasons, and until an accurate complexity analysis is available, we choose to attain to the latest measurable guidelines and choose our parameters such that the dimension of the solution space for the algebraic system is strictly greater than 20. We then compute the bound given by equation (5.1) and report it as an additional indication of the expected complexity of the attack. We hope that this work will encourage further study into FOPT and folding attacks in relation to GS codes.

**Norm-trace codes.** An attack based on *norm-trace codes* has been recently introduced by Barelli and Couvreur [4]. As the name suggests, these codes are the result of the application of both the trace and the norm operation to a certain support vector, and they are alternant codes. In particular, they are subfield subcodes of Reed–Solomon codes. The construction of these codes is given explicitly only for the specific case $m = 2$ (as will be the case in all DAGS parameters), i.e. the support vector has components in $\mathbb{F}_{q^2}$, in which case the norm-trace code is defined as

$$\mathcal{NT}(\boldsymbol{x}) = \langle \mathbf{1}, \mathrm{Tr}(\boldsymbol{x}), \mathrm{Tr}(\alpha \boldsymbol{x}), \mathrm{N}(\boldsymbol{x}) \rangle,$$

where $\alpha$ is an element of trace 1.

The main idea of the attack is that there exists a specific norm-trace code that is the *conductor* of the secret subcode into the public code. By "conductor" the authors refer to the largest code for which the Schur product (i.e. the component-wise product of all codewords, denoted by $\star$) is entirely contained in the target, i.e.

$$\mathrm{Cond}(\mathcal{D}, \mathcal{C}) = \{\boldsymbol{u} \in \mathbb{F}_q^n : \boldsymbol{u} \star \mathcal{D} \subseteq \mathcal{C}\}.$$

The authors present two strategies to determine the secret subcode. The first strategy is essentially an exhaustive search over all the codes of the proper co-dimension. This co-dimension is given by $2q/s$, since $s$ is the size of the permutation group of the code, which is non-trivial in our case due to the code being quasi-dyadic.

| Security level* | $q$ | $m$ | $n$ | $k$ | $k'$ | $s$ | $t$ | $w$ | Attack |
|---|---|---|---|---|---|---|---|---|---|
| 1 | $2^5$ | 2 | 832 | 416 | 43 | $2^4$ | 13 | 104 | $2^{70}$ |
| 3 | $2^6$ | 2 | 1216 | 512 | 43 | $2^5$ | 11 | 176 | $2^{80}$ |
| 5 | $2^6$ | 2 | 2112 | 704 | 43 | $2^6$ | 11 | 352 | $2^{55}$ |

**Table 3:** Early DAGS parameters. * Claimed.

While such a brute force in principle would be too expensive, the authors present a few refinements that make it feasible, which include an observation on the code rate of the codes in use, and the use of shortened codes.

The second strategy, instead, consists of solving a bilinear system, which is obtained using the parity-check matrix of the public code and treating as unknowns the elements of a generator matrix for the secret code (as well as the support vector $\boldsymbol{x}$). This system is solved using Gröbner bases techniques, and benefits from a reduction in the number of variables similar to the one performed in FOPT, as well as the refinements mentioned above (shortened codes).

In any case, it is easy to deduce that the two parameters $q$ and $s$ are crucial in determining the cost of running this step of the attack, which dominates the overall cost. In fact, the authors are able to provide an accurate complexity analysis for the first strategy which confirms this intuition. The average number of iterations of the brute force search is given by $q^{2c}$, where $c$ is exactly the co-dimension described above, i.e. $c = 2q/s$. In addition, it is shown that the cost of computing Schur products is $2n^3$ operations in the base field. Thus, the overall cost of the recovery step is $2n^3q^{4q/s}$ operations in $\mathbb{F}_q$. The authors then argue that wrapping up the attack has negligible cost, and that $q$-ary operations can be done in constant time (using tables) when $q$ is not too big. All this leads to a complexity which is below the desired security level for all of the DAGS parameters that had been proposed at the time of submission. We report these numbers in Table 3.

As it is possible to observe, the attack complexity is especially low for the last set of parameters since the dyadic order $s$ was chosen to be $2^6$, and this is probably too much to provide security against this attack. Still, we point out that, at the time this parameters were proposed, there was no indication this was the case, since this attack is using an entirely new technique, and it is unrelated to the FOPT and folding attacks that we just described.

Unfortunately, the attack authors were not able to provide a security analysis for the second strategy (bilinear system). This is due to the fact that the attack uses Gröbner based techniques, and it is very hard to evaluate the cost in this case (similarly to what happened for FOPT). In this case then, the only evidence the authors provide is experimental, and based on running the attack in practice on all the parameters. The authors report running times around 15 minutes for the first set and less than a minute for the last, while they admit they were not able to complete the execution in the middle case. This seems to match the evidence from the complexity results obtained for the first strategy, and suggests a speedup proportional to those. Further test runs are currently planned, but the fact that the attack already fails to run in practice for the middle set, gives us some confidence to believe that updated parameters will definitely make the attack infeasible.

## 5.4 Parameter selection

To choose our parameters, we keep in mind all the remarks from the previous sections about decoding attacks and structural attacks. As we have just seen, we need to respect the condition $mt \geq 21$ to guarantee security against FOPT. At the same time, to prevent the BC attack $q$ has to be chosen large enough and $s$ cannot be too big. Finally, for ISD to be computationally intensive, we require a sufficiently large number $w$ of errors to decode. This is given by $st/2$ according to the minimum distance of GS codes.

In addition, we tune our parameters to optimize performance. In this regard, the best results are obtained when the extension degree $m$ is as small as possible. This, however, requires the base field to be large enough to accommodate sufficiently big codes (against ISD attacks), since the maximum size for the code length $n$ is capped by $q^m - s$. Realistically, this means we want $q^m$ to be at least $2^{12}$, and the optimal choice in this sense

| Security level | $q$ | $m$ | $n$ | $k$ | $k'$ | $s$ | $t$ | $w$ | $n_{Y'}$ | BC |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $2^6$ | 2 | 832 | 416 | 43 | $2^4$ | 13 | 104 | 25 | $\approx 2^{128}$ |
| 3 | $2^8$ | 2 | 1216 | 512 | 32 | $2^5$ | 11 | 176 | 21 | $\approx 2^{288}$ |
| 5 | $2^8$ | 2 | 1600 | 896 | 32 | $2^5$ | 11 | 176 | 21 | $\approx 2^{289}$ |

**Table 4:** Suggested DAGS parameters.

seems to be $q = 2^8$ (see Section 6). Finally, note that $s$ is constrained to be a power of 2, and that odd values of $t$ seem to offer best performance.

Putting all the pieces together, we are able to present three sets of parameters, in Table 4. These correspond to three of the security levels indicated by NIST (first column), which are related to the hardness of performing a key search attack on three different variants of a block cipher, such as AES (with key length respectively 128, 192 and 256). As far as quantum attacks are concerned, we claim that ISD with Grover (see above) will usually require more resources than a Grover search attack on AES for the circuit depths suggested by NIST (parameter MAXDEPTH). Thus, classical security bits are the bottleneck in our case, and as such we choose our parameters to provide 128, 192 and 256 bits of classical security for security levels 1, 3 and 5 respectively. For practical reasons, during the rest of the paper we will refer to these parameters respectively as DAGS_1, DAGS_3 and DAGS_5.

For the above parameters, it is easy to observe that the value $n_{Y'}$ is always greater or equal to 21 (it is in fact 25 for DAGS_1), which keeps us clear of FOPT. With respect to the BC attack, the complexity analysis provided by the authors results in a value of $\approx 2^{126}$ for DAGS_1 and more than $2^{288}$ for the other two sets. Finally, the running cost of ISD (using Peters' script) is estimated at $2^{128}$, $2^{192}$ and $2^{256}$ respectively, as desired.

# 6 Performance analysis

## 6.1 Components

DAGS operates on vectors of elements of the finite field $\mathbb{F}_q$, where $q$ is a power of 2 as specified by the choice of parameters. Finite field elements are represented as bit strings using standard log/antilog tables (see for instance [29, Chapter 4, Section 5]) which are stored in the memory.

For DAGS_1, the finite field $\mathbb{F}_{2^6}$ is built using the polynomial $x^6 + x + 1$ and then extended to $\mathbb{F}_{2^{12}}$ using the quadratic irreducible polynomial $x^2 + \alpha x + \alpha$, where $\alpha$ is a primitive element of $\mathbb{F}_{2^6}$. In particular, using a well-known result on finite fields, we choose $\alpha = \gamma^{65}$ where $\gamma$ is a primitive element of $\mathbb{F}_{2^{12}}$. This particular choice allows for more efficient arithmetic using a *conversion matrix* to switch between different field representations. Similarly, for DAGS_3 and DAGS_5, we build the base field using $x^8 + x^4 + x^3 + x^2 + 1$ and the extension field $\mathbb{F}_{2^{16}}$ is obtained via $x^2 + \beta^{50}x + \beta$, where $\beta$ is a primitive element of $\mathbb{F}_{2^8}$.

## 6.2 Randomness generation

The randomness used in our implementation is provided by the NIST API. It uses AES as a PNGR, where NIST chooses the seed in order to have a controlled environment for tests. We use this random generator to obtain our input message $\boldsymbol{m}$, after which we calculate $\mathcal{G}(\boldsymbol{m})$ and $\mathcal{H}(\boldsymbol{m})$, where $\mathcal{G}$ is an expansion function and $\mathcal{H}$ is a compression function. In practice, we compute both using the *KangarooTwelve* function [48] from the Keccak family. To generate a low-weight error vector, we take part of $\mathcal{G}(\boldsymbol{m})$ as a seed $\boldsymbol{\sigma}$. We use again *KangarooTwelve* to expand the seed into a string of length $n$, then transform the latter into a fixed-weight string using a deterministic function.

## 6.3 Efficient private key reconstruction and decoding

As mentioned in Section 3, in our scheme we use a standard alternant decoder (step (2) of Algorithm 3), which requires the input to be a matrix in alternant form, i.e. $H'_{ij} = y_j x_j^i$ for $i = 0, \ldots, st-1$ and $j = 0, \ldots, n-1$. The first step consists of computing the syndrome of the received word, $H'\boldsymbol{c}^T$. Now, defining the whole alternant matrix $H'$ as private key would require storing $stn$ elements of $\mathbb{F}_{q^m}$, leading to huge key sizes. It would be possible to store as private key just the defining vectors $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{z}$, and then compute the alternant form during decapsulation. Doing so would drastically reduce the private key size, but would also significantly slow down the decapsulation algorithm. Thus we came up with a neat idea, and implemented a hybrid approach. We use $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{z}$ to compute the vector $\boldsymbol{y}$ during key generation and store $(\boldsymbol{v}, \boldsymbol{y})$ as private key, which still results in a compact size. Then, we complete the computation of the alternant form in the decapsulation algorithm. To avoid an unnecessary overhead, we incorporate this computation together with the syndrome computation. The process is detailed as follows.

**Algorithm 4** (Alternant-Syndrome Computation).
(1) Input received word $\boldsymbol{c}$ to be decoded.
(2) Compute the vector $\boldsymbol{s} = \mathrm{Diag}(\boldsymbol{y}) \cdot \boldsymbol{c}^T$.
(3) Form intermediate matrix $\tilde{H}$. To do this:
   (a) Set first row equal to $\boldsymbol{s}$.
   (b) Obtain row $i$, for $i = 1, \ldots, st-1$, by multiplying the $j$-th element of row $i-1$ by $v_j$, for $j = 0, \ldots, n-1$.
(4) Sum elements in each row and output resulting vector.

## 6.4 Measurements

The implementation is in ANSI C, as requested for a generic reference implementation. For the measurements, we used an x64 Intel Core i5-5300U processor at 2.30 GHz, 16 GiB of RAM and GCC version 6.3.0 20170516 without any optimization, running on Debian 9.2.

We start by considering space requirements. We recall the flow between two parties $\mathsf{P}_1$ and $\mathsf{P}_2$ in a generic key exchange protocol derived from a KEM:

$$
\begin{array}{ccc}
\mathsf{P}_1 & & \mathsf{P}_2 \\
(pk, sk) \leftarrow \mathrm{KEM.KeyGen} & \xrightarrow{\quad pk \quad} & \\
 & & (\boldsymbol{k}, \boldsymbol{c}) \leftarrow \mathrm{KEM.Encaps}(pk) \\
\boldsymbol{k}/\perp \leftarrow \mathrm{KEM.Decaps}(\boldsymbol{c}, sk) & \xleftarrow{\quad \boldsymbol{c} \quad} & \\
\text{shared key} := \boldsymbol{k} & &
\end{array}
\tag{6.1}
$$

When instantiated with DAGS, the public key is given by the generator matrix $G$. The non-identity block $M^T$ is $k \times (n-k) = k \times mst$ and is dyadic of order $s$, thus requires only $kmst/s = kmt$ elements of the base field for storage. The private key is simply the pair $(\boldsymbol{v}, \boldsymbol{y})$, consisting of $2n$ elements of $\mathbb{F}_{q^m}$. Finally, the ciphertext is the pair $(\boldsymbol{c}, \boldsymbol{d})$, that is, a $q$-ary vector of length $n$ plus 256 bits. This leads to the measurements (in bytes) given in Table 5 and Table 6.

| Parameter set | Public key | Private key | Ciphertext |
|---|---|---|---|
| DAGS_1 | 8112 | 2496 | 656 |
| DAGS_3 | 11264 | 4864 | 1248 |
| DAGS_5 | 19712 | 6400 | 1632 |

**Table 5:** Memory requirements.

| Message flow | Transmitted message | Size | | |
|---|---|---|---|---|
| | | DAGS_1 | DAGS_3 | DAGS_5 |
| $P_1 \rightarrow P_2$ | $G$ | 8112 | 11264 | 19712 |
| $P_2 \rightarrow P_1$ | $(\boldsymbol{c}, \boldsymbol{d})$ | 656 | 1248 | 1632 |

**Table 6:** Communication costs in protocol flow.

| | Cycles | | |
|---|---|---|---|
| Algorithm | DAGS_1 | DAGS_3 | DAGS_5 |
| Key Generation | 2540311986 | 4320206006 | 7371897084 |
| Encapsulation | 12108373 | 26048972 | 96929832 |
| Decapsulation | 215710551 | 463849016 | 1150831538 |

**Table 7:** Timings.

We now move on to analyzing time measurements. We are using x64 architecture and our measurements use an assembly instruction to get the time counter. We do this by calling "rdtsc" before and after the instruction, which gives us the cycles used by each function. Table 7 gives the results of our measurements represented by the mean after running the code 50 times.

## 6.5 Comparison

We thought it useful to provide a comparison with other recently proposed code-based KEMs (and in particular, NIST submissions). In Table 8, we present data for Classic McEliece, BIKE and BIG QUAKE with regards to memory requirements, for the highest security level (256 bits classical). We did not deem necessary, on the other hand, to provide a comparison in terms of implementation timings, as reference implementations are designed for clarity, rather than performance.

It is easy to see that the public key is much smaller than Classic McEliece and BIG QUAKE, and similar to that of BIKE. With regards to the latter, note that, for the same security level, the total communication bandwidth is of the same order of magnitude. This is because DAGS uses much shorter codes, and as a consequence the ciphertext is considerably smaller than a BIKE ciphertext. Moreover, for the purposes of a fair comparison, we remark that BIKE uses ephemeral keys, has a non-negligible decoding failure rate and only claims IND-CPA security – all factors that can restrict its use in various applications.

| Parameter set | Public key | Private key | Ciphertext |
|---|---|---|---|
| Classic McEliece | 1047319 | 13908 | 226 |
| BIKE-1 | 8187 | 548 | 8187 |
| BIKE-2 | 4093 | 548 | 4093 |
| BIKE-3 | 9032 | 565 | 9032 |
| BIG QUAKE | 149800 | 41804 | 492 |
| DAGS_5 | 19712 | 6400 | 1632 |

**Table 8:** Comparison of code-based KEMs (bytes).

# 7 Conclusion

In this paper, we presented DAGS, a key encapsulation mechanism based on quasi-dyadic generalized Srivastava codes. We proved that DAGS is IND-CCA secure in the random oracle model, and in the quantum random oracle model. Thanks to this feature, it is possible to employ DAGS not only as a key exchange protocol (for which IND-CPA would be a sufficient requirement), but also in other contexts such as hybrid encryption, where IND-CCA is of paramount importance.

In terms of performance, DAGS compares well with other code-based protocols, as shown by Table 8 and the related discussion (above). Another advantage of our proposal is that it does not involve any decoding error. This is particularly favorable in a comparison with some lattice-based schemes like [15], [2] and [14], as well as BIKE. No decoding error allows for a simpler formulation and better security bounds in the IND-CCA security proof.

Unlike traditional code-based protocols, DAGS features small sizes for all components, that is ciphertexts, private keys (thanks to our improved computation idea) and public keys. All the objects involved in the computations are vectors of finite fields elements, which in turn are represented as binary strings; thus computations are fast. The cost of computing the hash functions is minimized thanks to the parameter choice that makes sure the input $\boldsymbol{m}$ is only 256 bits. As a result, we expect our scheme to be implemented efficiently on multiple platforms.

The current reference code for the scheme is available at the repository https://git.dags-project.org/dags/dags. Our team is currently at work to complete various implementations that can better showcase the potential of DAGS in terms of performance. These include code prepared with x86 assembly instructions (AVX) as well as a hardware implementation (FPGA) etc. A hint at the effectiveness of DAGS can be had by looking at the performance of the scheme presented in [16], which also features an implementation for embedded devices. In particular, we expect DAGS to perform especially well in hardware, due to the nature of the computations of the McEliece framework.

Finally, we would like to highlight that a DAGS-based key exchange features an "asymmetric" structure, where the bandwidth cost and computational effort of the two parties are considerably different. In particular, in the flow described in (6.1), the party $\mathsf{P}_2$ benefits from a much smaller message and faster computation (the encapsulation operation), whereas $\mathsf{P}_1$ has to perform a key generation and a decapsulation (which includes a run of the decoding algorithm), and transmit a larger message (the public matrix). This is suitable for traditional client-server applications where the server side is usually expected to respond to a large number of requests and thus benefits from a lighter computational load. On the other hand, it is easy to imagine an instantiation, with reversed roles, which could be suitable for example in Internet-of-Things (IoT) applications, where it would be beneficial to lesser the burden on the client side, due to its typical processing, memory and energy constraints. All in all, our scheme offers great flexibility in key exchange applications, which is not the case for traditional key exchange protocols like Diffie–Hellman.

In light of all these aspects, we believe that DAGS is a promising candidate for post-quantum cryptography standardization as a key encapsulation mechanism.

# A Note on the choice of $\omega$

As discussed in Section 6.3, in our scheme we use a standard alternant decoder. After computing the syndrome of the word to be decoded, the next step is to recover the error locator polynomial $\sigma(x)$, by means of the euclidean algorithm for polynomial division; the algorithm then proceeds by finding the roots of $\sigma$. There is a one-to-one correspondence between these roots and the error positions: in fact, there is an error in position $i$ if and only if $\sigma(1/x_i) = 0$.

Of course, if one of the $x_i$'s is equal to 0, it is not possible to find the root, and to detect the error.

Now, the generation of the error vector is random, hence we can assume the probability of having an error in position $i$ to be around $st/2n$; since the codes give the best performance when $mst$ is close to $n/2$, we can

estimate this probability as $1/4m$, which is reasonably low for any non-trivial choice of $m$; however, we still argue that the code is not fully decodable and we now explain how to adapt the key generation algorithm to ensure that all the $x_i$'s are non-zero.

As part of the key generation algorithm we assign to each $x_i$ the value $v_i$, hence it is enough to restrict the possible choices for $\omega$ to the set $\{\alpha \in \mathbb{F}_{q^m} \mid \alpha \neq 1/h_i + 1/h_0,\ i = 0, \ldots, n-1\}$. In doing so, we considerably restrict the possible choices for $\omega$ but we ensure that the decoding algorithm works properly.

## References


[1] A. Al Jabri, A statistical decoding algorithm for general linear block codes, in: *Cryptography and Coding*, Lecture Notes in Comput. Sci. 2260, Springer, Berlin (2001), 1–8.

[2] E. Alkim, L. Ducas, T. Pöppelmann and P. Schwabe, Post-quantum key exchange - a new hope, Cryptology ePrint Archive Report 2015/1092 (2015), http://eprint.iacr.org/2015/1092.

[3] M. Baldi, F. Chiaraluce, R. Garello and F. Mininni, Quasi-cyclic low-density parity-check codes in the McEliece cryptosystem, in: *IEEE International Conference on Communications—ICC'07*, IEEE Press, Piscataway (2007), 951–956.

[4] E. Barelli and A. Couvreur, An efficient structural attack on nist submission dags, preprint (2018), https://arxiv.org/abs/1805.05429.

[5] S. Barg, Some new NP-complete coding problems (in Russian), *Problemy Peredachi Informatsii* **30** (1994), no. 3, 23–28.

[6] A. Barg, Complexity issues in coding theory, in: *Handbook of Coding Theory. Vol. 1. Part 1: Algebraic Coding*, Elsevier, Amsterdam (1998), 649–754.

[7] P. S. L. M. Barreto, S. Gueron, T. Gueneysu, R. Misoczki, E. Persichetti, N. Sendrier and J.-P. Tillich, Cake: Code-based algorithm for key encapsulation, in: *Cryptography and Coding—IMACC 2017*, Springer, Cham (2017), 207–226.

[8] P. S. L. M. Barreto, R. Lindner and R. Misoczki, Monoidic codes in cryptography, in: *Post-quantum Cryptography*, Lecture Notes in Comput. Sci. 7071, Springer, Heidelberg (2011), 179–199.

[9] T. P. Berger, P.-L. Cayrel, P. Gaborit and A. Otmani, Reducing key length of the McEliece cryptosystem, in: *Progress in Cryptology—AFRICACRYPT 2009*, Lecture Notes in Comput. Sci. 5580, Springer, Berlin (2009), 77–97.

[10] E. R. Berlekamp, R. J. McEliece and H. C. A. van Tilborg, On the inherent intractability of certain coding problems, *IEEE Trans. Inform. Theory* **IT-24** (1978), no. 3, 384–386.

[11] D. J. Bernstein, Grover vs. McEliece, in: *Post-Quantum Cryptography*, Lecture Notes in Comput. Sci. 6061, Springer, Berlin (2010), 73–80.

[12] D. J. Bernstein, T. Chou and P. Schwabe, Mcbits: Fast constant-time code-based cryptography, in: *Cryptographic Hardware and Embedded Systems—CHES 2013*, Lecture Notes in Comput. Sci. 8086, Springer, Berlin (2013), 250–272.

[13] B. Biswas and N. Sendrier, McEliece cryptosystem implementation: Theory and practice, in: *Post-quantum Cryptography*, Lecture Notes in Comput. Sci. 5299, Springer, Berlin (2008), 47–62.

[14] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan and D. Stebila, Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE, Cryptology ePrint Archive Report 2016/659 (2016), http://eprint.iacr.org/2016/659.

[15] J. W. Bos, C. Costello, M. Naehrig and D. Stebila, Post-quantum key exchange for the tls protocol from the ring learning with errors problem, in: *IEEE Symposium on Security and Privacy*, IEEE Press, Piscataway (2015), 553–570.

[16] P.-L. Cayrel, G. Hoffmann and E. Persichetti, Efficient implementation of a CCA2-secure variant of McEliece using generalized Srivastava codes, in: *Public Key Cryptography—PKC 2012*, Lecture Notes in Comput. Sci. 7293, Springer, Heidelberg (2012), 138–155.

[17] N. T. Courtois, M. Finiasz and N. Sendrier, How to achieve a McEliece-based digital signature scheme, in: *Advances in Cryptology—ASIACRYPT 2001*, Lecture Notes in Comput. Sci. 2248, Springer, Berlin (2001), 157–174.

[18] R. Cramer and V. Shoup, Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack, *SIAM J. Comput.* **33** (2003), no. 1, 167–226.

[19] J.-C. Deneuville, P. Gaborit and G. Zémor, Ouroboros: A simple, secure and efficient key exchange protocol based on coding theory, in: *Post-quantum Cryptography*, Lecture Notes in Comput. Sci. 10346, Springer, Cham (2017), 18–34.

[20] J.-C. Faugère, V. Gauthier-Umaña, A. Otmani, L. Perret and J.-P. Tillich, A distinguisher for high-rate McEliece cryptosystems, *IEEE Trans. Inform. Theory* **59** (2013), no. 10, 6830–6844.

[21] J.-C. Faugère, A. Otmani, L. Perret, F. de Portzamparc and J.-P. Tillich, Structural cryptanalysis of McEliece schemes with compact keys, *Des. Codes Cryptogr.* **79** (2016), no. 1, 87–112.

[22] 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*, Lecture Notes in Comput. Sci. 6110, Springer, Berlin (2010), 279–298.

[23] J.-C. Faugère, A. Otmani, L. Perret and J.-P. Tillich, Algebraic cryptanalysis of McEliece variants with compact keys – towards a complexity analysis, in: *Proceedings of the 2nd International Conference on Symbolic Computation and Cryptography—SCC'10*, Laboratoire d'Informatique de Paris 6, Paris (2010), 45–55.
[24] E. Fujisaki and T. Okamoto, Secure integration of asymmetric and symmetric encryption schemes, *J. Cryptology* **26** (2013), no. 1, 80–101.
[25] Q. Guo, T. Johansson and P. Stankovski, A key recovery attack on MDPC with CCA security using decoding errors, in: *Advances in Cryptology—ASIACRYPT 2016. Part I*, Lecture Notes in Comput. Sci. 10031, Springer, Berlin (2016), 789–815.
[26] Y. Hamdaoui and N. Sendrier, A non asymptotic analysis of information set decoding, Cryptology ePrint Archive Report 2013/162 (2013), http://eprint.iacr.org/2013/162.
[27] D. Hofheinz, K. Hövelmanns and E. Kiltz, A modular analysis of the Fujisaki–Okamoto transformation, in: *Theory of Cryptography. Part I*, Lecture Notes in Comput. Sci. 10677, Springer, Cham (2017), 341–371.
[28] G. Kachigar and J.-P. Tillich, Quantum information set decoding algorithms, in: *Post-quantum Cryptography*, Lecture Notes in Comput. Sci. 10346, Springer, Cham (2017), 69–89.
[29] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes. I*, North-Holland Math. Libr. 16, North-Holland, Amsterdam, 1977,
[30] R. J. McEliece, A public-key cryptosystem based on algebraic coding theory, *Deep Space Netw. Prog. Rep.* **44** (1978), 114–116.
[31] R. Misoczki and P. S. L. M. Barreto, Compact mceliece keys from goppa codes, in: *Selected Areas in Cryptography*, Springer, Berlin (2009), 376–392.
[32] R. Misoczki, J.-P. Tillich, N. Sendrier and P. L. S. M. Barreto, MDPC-McEliece: New McEliece variants from moderate density parity-check codes, in: *International Symposium on Information Theory—ISIT 2013*, IEEE Press, Piscataway (2013), 2069–2073.
[33] R. Niebuhr, Statistical decoding of codes over $\mathbb{F}_q$, in: *Post-quantum Cryptography*, Lecture Notes in Comput. Sci. 7071, Springer, Heidelberg (2011), 217–227.
[34] R. Niebuhr, E. Persichetti, P.-L. Cayrel, S. Bulygin and J. Buchmann, On lower bounds for information set decoding over $\mathbb{F}_q$ and on the effect of partial knowledge, *Int. J. Inf. Coding Theory* **4** (2017), no. 1, 47–78.
[35] R. Nojima, H. Imai, K. Kobara and K. Morozov, Semantic security for the McEliece cryptosystem without random oracles, *Des. Codes Cryptogr.* **49** (2008), no. 1–3, 289–305.
[36] E. Persichetti, Compact McEliece keys based on quasi-dyadic Srivastava codes, *J. Math. Cryptol.* **6** (2012), no. 2, 149–169.
[37] E. Persichetti, Secure and anonymous hybrid encryption from coding theory, in: *Post-Quantum Cryptography—PQCrypto 2013*, Berlin, Heidelberg (2013), 174–187.
[38] C. Peters, Information-set decoding for linear codes over $\mathbf{F}_q$, in: *Post-quantum Cryptography*, Lecture Notes in Comput. Sci. 6061, Springer, Berlin (2010), 81–94.
[39] E. Prange, The use of information sets in decoding cyclic codes, *IRE Trans.* **IT-8** (1962), S5–S9.
[40] D. V. Sarwate, On the complexity of decoding Goppa codes, *IEEE Trans. Inform. Theory* **IT-23** (1977), no. 4, 515–516.
[41] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.* **26** (1997), no. 5, 1484–1509.
[42] F. Strenzke, A timing attack against the secret permutation in the McEliece PKC, in: *Post-quantum Cryptography*, Lecture Notes in Comput. Sci. 6061, Springer, Berlin (2010), 95–107.
[43] F. Strenzke, E. Tews, H. G. Molter, R. Overbeck and A. Shoufan, Side channels in the McEliece PKC, in: *Post-quantum Cryptography*, Lecture Notes in Comput. Sci. 5299, Springer, Berlin (2008), 216–229.
[44] https://bigquake.inria.fr/.
[45] https://bikesuite.org.
[46] http://christianepeters.wordpress.com/publications/tools/.
[47] https://classic.mceliece.org/.
[48] https://keccak.team/kangarootwelve.html.