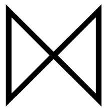
DAGS: Key Encapsulation using Dyadic GS Codes

Anonymized for Submission



Abstract. Code-based Cryptography is one of the main areas of interest for the Post-Quantum Cryptography Standardization call. In this paper, we introduce DAGS[1], 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.

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. NIST's call for papers for post-quantum standardization is a further reassurance about the need for solid post-quantum proposals.

Code-based cryptography is one of the main candidates for this task. The area is based on the Syndrome Decoding Problem [11], which shows no vulnerabilities to quantum attacks. Over the years, since McEliece's seminal work [32], many cryptosystems have been proposed, trying to

[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.

balance security and efficiency and in particular dealing with inherent flaws such as the large size of the public keys. In fact, while McEliece's cryptosystem (based on binary Goppa codes) is still formally unbroken, it features a key of several 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 such as Goppa or Srivastava codes; the second suggests to use sparse matrices as in LDPC/MDPC codes. This work builds on the former approach, initiated in 2009 by Berger et al. [10], who proposed Quasi-Cyclic (QC) codes, and Misoczki and Barreto [33], suggesting Quasi-Dyadic (QD) codes instead (later generalized to Quasi-Monoidic (QM) codes [9]). 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, Otmani, Perret and Tillich [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 definitely 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 [37].

Our Contribution. In this paper, we present DAGS, a Key Encapsulation Mechanism (KEM) 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 [19]. To the best of our knowledge, this is the first code-based KEM that uses quasi-dyadic codes. Another NIST submission, named BIG QUAKE [2], proposes a scheme based on quasi-cyclic codes.
Our KEM achieves IND-CCA security following the recent framework by Kiltz et al. [28], 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 [4], as well as more recent proposals such as BIKE [3] and the aforementioned BIG QUAKE. The "Classic McEliece" project is an evolution of the well-known McBits [13](based on the work of Persichetti [38]), 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 [8] and Ouroboros [20] . The scheme possesses some 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) [26], 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 [10], and proposes to use quasi-cyclic Goppa codes. Due to the particular nature of the FOPT attack and its successors [24], 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 [5] and Frodo [15], 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. This 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. Section 5 features a discussion about practical security, including general decoding attacks (ISD) and the FOPT attack, and presents 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
A	a set
$\mathrm{Diag}(\mathbf{a})$	the diagonal matrix formed by the vector $\mathbf{a}$
I_n	the $n \times n$ identity matrix
$\xleftarrow{\$}$	choosing a random element from a set or distribution

2.2 Linear Codes

The *Hamming weight* of a vector $\boldsymbol{x} \in \mathbb{F}_q^n$ is given by the number $\mathsf{wt}(\boldsymbol{x})$ of its nonzero components. We define a linear code using the metric induced by the Hamming weight.

Definition 1. *An* (n, k)-linear code 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. *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,* $\Delta(t, \boldsymbol{h})$ *denotes the matrix* $\Delta(\boldsymbol{h})$ *truncated to its first* t *rows. Finally, we call a matrix quasi-dyadic if it is a block matrix whose component blocks are* $t \times t$ *dyadic submatrices.*

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

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

where each block is a $2^{k-1} \times 2^{k-1}$ dyadic matrix (and where any 1×1 matrix is dyadic).

Definition 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 nonzero elements of* $\mathbb{F}_{q^m}$*. The* Generalized Srivastava (GS) 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

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

The parameters for such a code are the length $n \leq q^m - s$, dimension $k \geq n - mst$ and minimum distance $d \geq st + 1$. GS codes are part of the family of Alternant codes, and therefore benefit of an efficient decoding algorithm. 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 [31, Ch. 12, §6].

3 DAGS

In this section we introduce the three algorithms that form DAGS – a key-encapsulation mechanism based on Quasi-Dyadic GS codes. 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.

The key generation process uses the following fundamental equation

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

to build the signature $\boldsymbol{h} = (h_0, \ldots, h_{n-1})$. This is then used to form a Cauchy matrix, i.e. a matrix $C(\boldsymbol{u}, \boldsymbol{v})$ with components $C_{ij} = \frac{1}{u_i - v_j}$. The matrix is then successively powered (element by element) forming several blocks which are superimposed and then multiplied by a random diagonal matrix. It is easy to see that this matrix is equivalent to a parity-check matrix for a GS code (by a row permutation) as described in Definition 3, 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 and row-reduced to systematic form to form the public key.

The process is described in detail below: note that this is essentially the same as in [37], to which we refer the reader for more details about dyadic GS codes and the key generation process.

Algorithm 1. Key Generation

```
1. Generate dyadic signature $\boldsymbol{h}$. To do this:
   (a) Choose random non-zero distinct $h_0$ and $h_j$ for $j = 2^l, l = 0, \dots, \lfloor \log q^m \rfloor$.
   (b) Form the remaining elements using (1).
   (c) Return a selection[2] of blocks of dimension $s$ up to length $n$.
2. Build the Cauchy support. To do this:
   (a) Choose a random[3] offset $\omega \xleftarrow{\$} \mathbb{F}_{q^m}$.
   (b) Set $u_i = 1/h_i + \omega$ and $v_j = 1/h_j + 1/h_0 + \omega$ for $i = 0, \dots, s-1$ and $j = 0, \dots, n-1$.
   (c) Set $\boldsymbol{u} = (u_0, \dots, u_{s-1})$ and $\boldsymbol{v} = (v_0, \dots, v_{n-1})$.
3. Form Cauchy matrix $\hat{H}_1 = C(\boldsymbol{u}, \boldsymbol{v})$.
4. Build blocks $\hat{H}_i$, $i = 1, \dots t$, by raising each element of $\hat{H}_1$ to the power of $i$.
5. Superimpose blocks to form matrix $\hat{H}$.
6. Choose random elements $z_i \xleftarrow{\$} \mathbb{F}_{q^m}$ with the restriction $z_{is+j} = z_{is}$ for $i = 0, \dots, n_0 - 1$, $j = 0, \dots, s-1$.
7. Form $H = \hat{H} \cdot \mathrm{Diag}(\boldsymbol{z})$.
8. Transform $H$ into alternant form[4]: call this $H'$.
9. Project $H$ onto $\mathbb{F}_q$ using the co-trace function: call this $H_{base}$.
10. Write $H_{base}$ 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 alternant matrix $H'$.
```

The encapsulation and decapsulation algorithms make use of two hash functions $\mathcal{G} : \mathbb{F}_q^{k'} \to \mathbb{F}_q^k$ and $\mathcal{H} : \mathbb{F}_q^{k'} \to \mathbb{F}_q^{k'}$, the former with the task of generating randomness for the scheme, the latter to provide "plaintext confirmation" as in [28]. The shared symmetric key is obtained via another hash function $\mathcal{K} : \{0,1\}^* \to \{0,1\}^\ell$, where ℓ is the desired key length.

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

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

[4] See §2 and §6 of [31, Ch. 12].

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 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 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 [31, Ch. 12, §9] and requires the parity-check matrix to be in alternant form (hence the nature of the private key).

Algorithm 3. Decapsulation

1. Input private key, i.e. parity-check matrix H' in alternant form.
2. Use H' to decode $\boldsymbol{c}$ and obtain codeword $\boldsymbol{\mu}' G$ and error $\boldsymbol{e}'$.
3. Output $\perp$ if decoding fails or $\mathsf{wt}(\boldsymbol{e}') \neq w$
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 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 encryption framework, with a notable exception. In fact, we incorporate the "randomized" version of McEliece by Nojima et al. [36] 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 [28], in the fact that we don't perform a full re-encryption 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. *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 $\mathsf{Decaps}(sk, \boldsymbol{c})$.*
3. *Query an encryption oracle. The oracle runs $\mathsf{Encaps}(pk)$ 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 ℓ 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*

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

We say that a KEM is secure if the advantage $\mathsf{Adv}_{KEM}^{IND-CCA}$ 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 [36]) satisfies a simple property. This will allow us to get better security bounds in our reduction.

Definition 5. *Consider a probabilistic PKE with randomness set* R. *We say that PKE is* γ-spread *if for a given key pair* (sk, pk)*, a plaintext* m *and an element* y *in the ciphertext domain, we have*

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

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

The definition above is presented as in [28], but note that in fact this corresponds to the notion of γ-*uniformity* given by Fujisaki and Okamoto in [25], except for a change of constants. In other words, a scheme is γ-spread if it is $2^{-\gamma}$-uniform.

It was proved in [17] that a simple variant of the (classic) McEliece PKE is γ-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 1. *Randomized McEliece is* γ*-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 isn't. If it isn't, 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. $\square$

We are now ready to present the security results.

Theorem 1. *Let* $\mathcal{A}$ *be an IND-CCA adversary against DAGS that makes at most* $q_{RO} = q_{\mathcal{G}} + q_{\mathcal{K}}$ *total random oracle queries*[5] *and* q_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*

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

[5] Resp. $q_{\mathcal{G}}$ queries to the random oracle $\mathcal{G}$ and $q_{\mathcal{K}}$ queries to the random oracle $\mathcal{K}$.

Proof. The thesis is a consequence of the results presented in Section 3.3 of [28]. In fact, our scheme follows the $\mathsf{KEM}_m^{\perp}$ 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 a 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 δ in [28]), 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 2. *Let $\mathcal{A}$ be a quantum IND-CCA adversary against DAGS that makes at most $q_{RO} = q_{\mathcal{G}} + q_{\mathcal{K}}$ total quantum random oracle queries*[6] *and q_D (classical) decryption queries. Then there exists a OW-CPA adversary $\mathcal{B}$ against PKE, running in approximately the same time as $\mathcal{A}$, such that*

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

Proof. The thesis is a consequence of the results presented in Section 4.4 of [28]. In fact, our scheme follows the $\mathsf{QKEM}_m^{\perp}$ framework that consists of applying two generic transformations to a public-key encryption scheme. The first step transforming the IND-CPA encryption scheme into a 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

[6] Same as in Theorem 1.

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. $\square$

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)*.

Problem 1. Given an $(n-k)\times n$ full-rank matrix H and a vector $\boldsymbol{y}$, both defined over $\mathbb{F}_q$, and a non-negative integer w, find a vector $\boldsymbol{e}\in\mathbb{F}_q^n$ of weight w such that $H\boldsymbol{e}^T=\boldsymbol{y}$.

The corresponding decision problem was proved to be NP-complete in 1978 [11], but only for binary codes. In 1994, A. Barg proved that this result holds for codes over all finite fields ([6], in Russian, and [7, 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. [21]) for cryptographic protocols that make use of high-rate Goppa codes (like the CFS signature scheme [18]). 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, [14]) 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 [40]. Among several variants and generalizations, Peters showed [39] that it is possible to apply Prange's approach to generic q-ary codes. Other approaches such as Statistical Decoding [29,34] are usually considered less efficient. Thus, when choosing parameters, we will focus mainly on defeating attacks of the ISD family.

Hamdaoui and Sendrier in [27] provide non-asymptotic complexity estimates for ISD in the binary case. For codes over $\mathbb{F}_q$, instead, a bound is given in [35], 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[1].

Quantum Speedup. Bernstein in [12] shows that Grover's algorithm applies to ISD-like algorithms, effectively halving the asymptotic exponent in the complexity estimates. Later, it was proven in [30] 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 FOPT

While, as we discussed above, recovering a private matrix from a public one can be in general a very difficult problem, the presence of extra structure in the code properties can have a considerable effect in lowering this difficulty.

A very effective structural attack was introduced by Faugère, Otmani, Perret and Tillich in [22]. The attack (for convenience referred to as FOPT) relies on the simple property (valid for every linear code)

$H \cdot G^T = 0$ to build an algebraic system, using then Gröbner bases techniques to solve it. The special properties of alternant codes are fundamental, as they contribute to considerably reduce the number of unknowns of the system.

The attack was originally aimed at two variants of McEliece, introduced respectively in [10] and [33]. The first variant, using quasi-cyclic codes, was easily broken in all proposed parameters, and falls out of the scope of this paper. The second variant, instead, only considered quasi-dyadic Goppa codes. In this case, most of the parameters proposed have also 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, as they were keeping constant the value $q^m = 2^{16}$. As it turns out, the extension degree m plays a key role in the attack, as it defines the dimension of the solution space, which is equal, in fact, exactly to $m - 1$. In a successive paper [23], the authors provide a theoretical complexity bound for the attack, and 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. In the case of GS codes, though, there is one important difference to keep in mind. In fact, as shown in [37], the dimension of the solution space is defined by $mt-1$, rather than $m-1$ as for Goppa codes. This provides greater flexibility when designing parameters for the code, and it allows, for example, to keep the extension degree m small.

Recently, an extension of the FOPT attack appeared in [24]. 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 much smaller code (the "folded" code), thanks to the strong properties of the automorphism group of the alternant codes in use. 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. The paper features two tables with several sets of parameters, respectively for signature schemes, and encryption schemes. The parameters are either taken from the original papers, or generated ad hoc. While codes designed to work for signature schemes turn out to be very easy to attack (due to their particular nature), the situation for encryption is more complex. Despite a refinement in the techniques used to solve the algebraic system, some of the parameters could not be solved in practice, and even the binary Goppa codes of [33], with their relatively low dimension of 15, require a considerably high computational effort (at least 2^{150} operations).

It is not clear how the attack performs against GS codes, since the authors didn't present any explicit result against this particular family of codes, nor attempted to decode GS codes specifically. Thus, an attack against GS codes would use generic techniques for Alternant codes, and wouldn't benefit from the speedups which are specific to (binary) Goppa codes. Furthermore, the authors do not propose a concrete bound, but only provide experimental results. For these reasons, and until an accurate complexity analysis for an attack on GS codes is available, we choose to attain to the latest measurable guidelines (those suggested in [23]) and choose our parameters such that the dimension of the solution space for the algebraic system is strictly greater than 20. We hope that this work will encourage further study into FOPT and folding attacks in relation to GS codes.

5.4 Parameter Selection

To choose our parameters, we have to first keep in mind all of the remarks from the previous sections about decoding and structural attacks. For FOPT, we have the condition $mt \geq 21$. This guarantees at least 128 bits of security according to the bound presented in [23]. On the other hand, 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 an optimal choice in this sense seems to be $q = 2^6, m = 2$. 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 set of parameters, in the following table. These correspond to three of the security levels indicated by NIST, 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 (first column).

We also included the estimated complexity of the structural attack (column FOPT), which is at least greater than 128 bits in all cases.

Table 1: Suggested DAGS Parameters.

Security Level	FOPT	q	m	n	k	k'	s	t	w
128	≥ 128	2^5	2	832	416	43	2^4	13	104
192	≥ 128	2^6	2	1216	512	43	2^5	11	176
256	≥ 128	2^6	2	2112	704	43	2^6	11	352

For practical reasons, during the rest of the paper we will refer to these parameters respectively as DAGS_1, DAGS_3 and DAGS_5.

6 Performance Analysis

6.1 Components

For DAGS_3 and DAGS_5, 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 $x^2 + x + \alpha^{34}$, where α is the primitive element of $\mathbb{F}_{2^6}$. For DAGS_1, we build $\mathbb{F}_{2^5}$ using $x^5 + x^2 + 1$ and then extend it to $\mathbb{F}_{2^{10}}$ via $x^2 + \alpha^4 x + \alpha$.

DAGS computations are detailed as follows:

Key generation:

1. Two polynomial multiplications in $\mathbb{F}_{q^m}$ and two in $\mathbb{F}_q$.
2. Six polynomial inversions in $\mathbb{F}_{q^m}$ and four in $\mathbb{F}_q$.
3. Two polynomial squarings in $\mathbb{F}_{q^m}$ and two in $\mathbb{F}_q$.
4. Two polynomial additions in $\mathbb{F}_{q^m}$ and two in $\mathbb{F}_q$.
5. One random generation of a polynomial in $\mathbb{F}_{q^m}$.

Encapsulation:

1. One polynomial multiplication in $\mathbb{F}_q$.
2. One polynomial addition in $\mathbb{F}_q$.
3. One random generation of a polynomial in $\mathbb{F}_q$.
4. One hash function computation.

Decapsulation:

1. Three polynomial multiplications in $\mathbb{F}_{q^m}$.
2. One polynomial power in $\mathbb{F}_{q^m}$.
3. One polynomial addition in $\mathbb{F}_{q^m}$.
4. One random generation of a polynomial in $\mathbb{F}_q$.
5. One hash function computation.

6.2 Measurements

In Table 2 we recall the flow between two parties P_1 and P_2 in a standard Key Exchange protocol derived from a KEM.

Table 2: KEM-based Key Exchange flow

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

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 given by the alternant matrix H' which is composed of stn 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 following measurements (in bytes).

Table 3: Memory Requirements.

Parameter Set	Public Key	Private Key	Ciphertext
DAGS_1	6760	432640	552
DAGS_3	8448	1284096	944
DAGS_5	11616	2230272	1616

Table 4: Communication Bandwidth.

Message Flow	Transmitted Message	Size		
		DAGS_1	DAGS_3	DAGS_5
$\mathsf{P}_1 \to \mathsf{P}_2$	G	6760	8448	11616
$\mathsf{P}_2 \to \mathsf{P}_1$	$(\mathbf{c}, \mathbf{d})$	552	944	1616

We would like to mention that the representation of the private key offers a significant tradeoff between time and space. In fact, it would be possible to store as private key just the defining vectors $\mathbf{u}, \mathbf{v}$ and $\mathbf{z}$, and then compute the alternant form (step 8. of the key generation algorithm) during decapsulation. Doing so would reduce the private key size to a few kilobytes[7], but would also significantly slow down the decapsulation algorithm. Thus, we have opted to store H' instead and save computation time, although this obviously results in a very large private key: this is generally the better option when using static keys, and in software implementations. However, in other situations (e.g. hardware) where private key size is relevant, the alternative representation might be preferable. Therefore we signal this as an implementor's choice.

6.3 Comparison

We thought it useful to provide a comparison with other recently-proposed code-based KEMs (and in particular, NIST submissions). In the following table, 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.

[7] Namely, $s + n + n_0$ elements of $\mathbb{F}_{q^m}$, leading to, approximately, 1, 2 and 3 Kb for DAGS_1, DAGS_3 and DAGS_5 respectively.

Table 5: Memory Requirements.

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	11616	2230272	1616

It is easy to see that the public key is much smaller than Classic McEliece and BIG QUAKE, and of the same order of magnitude of BIKE. With regards to the latter, note that, for the same security level, the total communication bandwidth is also of the same order of magnitude, and it is in fact even smaller for the case of BIKE-1 and BIKE-3. This is because, while the size of the public key is slightly less than a DAGS key, 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. It is also evident that our choice of private key currently gives a much larger size than all the other schemes. However, we have discussed above how, should it be necessary, the size of the private key can be greatly reduced, again to the same order of magnitude (just a bit larger) than a BIKE private key.

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 5 and the related discussion (above). Another advantage of our proposal is that it doesn't involve any decoding error. This is particularly favorable in a comparison with some lattice-based schemes like [16], [5] and [15], as well as BIKE. No decoding error allows for a simpler formulation and better security bounds in the IND-CCA security proof.

As is the case in most code-based protocols, 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 $\boldsymbol{\mu}$ 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 could 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 [17], 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 Table 2, the party P_2 benefits from a much smaller message and faster computation (the encapsulation operation), whereas 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.

References

1. http://christianepeters.wordpress.com/publications/tools/.
2. https://bigquake.inria.fr/.
3. https://bikesuite.org.
4. https://classic.mceliece.org/.
5. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - a new hope. Cryptology ePrint Archive, Report 2015/1092, 2015. http://eprint.iacr.org/2015/1092.
6. A. Barg. Some new NP-complete coding problems. *Probl. Peredachi Inf.*, 30:23–28, 1994. (in Russian).
7. A. Barg. Complexity issues in coding theory. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(46), 1997.
8. Paulo SLM Barreto, Shay Gueron, Tim Gueneysu, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, and Jean-Pierre Tillich. Cake: Code-based algorithm for key encapsulation.
9. Paulo SLM Barreto, Richard Lindner, and Rafael Misoczki. Monoidic codes in cryptography. *PQCrypto*, 7071:179–199, 2011.
10. T. P. Berger, P.-L. Cayrel, P. Gaborit, and A. Otmani. Reducing Key Length of the McEliece Cryptosystem. In *AFRICACRYPT*, pages 77–97, 2009.
11. E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *Information Theory, IEEE Transactions on*, 24(3):384 – 386, may 1978.
12. Daniel J. Bernstein. *Grover vs. McEliece*, pages 73–80. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
13. Daniel J. Bernstein, Tung Chou, and Peter Schwabe. Mcbits: Fast constant-time code-based cryptography. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8086 LNCS, pages 250–272, 12 2013.
14. B. Biswas and N. Sendrier. Mceliece cryptosystem implementation: Theory and practice. In *PQCrypto*, pages 47–62, 2008.
15. Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas 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.
16. Joppe W Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the tls protocol from the ring learning with errors problem. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 553–570. IEEE, 2015.
17. Pierre-Louis Cayrel, Gerhard Hoffmann, and Edoardo Persichetti. Efficient implementation of a cca2-secure variant of McEliece using generalized Srivastava codes. In *Proceedings of PKC 2012, LNCS 7293, Springer-Verlag*, pages 138–155, 2012.
18. N. Courtois, M. Finiasz, and N. Sendrier. How to achieve a mceliece-based digital signature scheme. In *ASIACRYPT*, pages 157–174, 2001.
19. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, January 2004.
20. Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Ouroboros: A simple, secure and efficient key exchange protocol based on coding theory. In Tanja Lange and Tsuyoshi Takagi, editors, *PQCrypto 2017*, volume 10346 of *LNCS*, pages 18–34. Springer, 2017.

21. J.-C. Faugère, V. Gauthier-Umaña, A. Otmani, L. Perret, and J.-P. Tillich. A distinguisher for high rate mceliece cryptosystems. In *Information Theory Workshop (ITW), 2011 IEEE*, pages 282 –286, oct. 2011.
22. J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic cryptanalysis of mceliece variants with compact keys. In *EUROCRYPT*, pages 279–298, 2010.
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 *SCC '10: Proceedings of the 2nd International Conference on Symbolic Computation and Cryptography*, pages 45–55, RHUL, June 2010.
24. Jean-Charles Faugere, Ayoub Otmani, Ludovic Perret, Frédéric De Portzamparc, and Jean-Pierre Tillich. Structural cryptanalysis of mceliece schemes with compact keys. *Designs, Codes and Cryptography*, 79(1):87–112, 2016.
25. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, volume 1666 of *LNCS*, pages 537–554. Springer, 1999.
26. Qian Guo, Thomas Johansson, and Paul Stankovski. *A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors*, pages 789–815. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.
27. Yann Hamdaoui and Nicolas Sendrier. A non asymptotic analysis of information set decoding. Cryptology ePrint Archive, Report 2013/162, 2013. http://eprint.iacr.org/2013/162.
28. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. Cryptology ePrint Archive, Report 2017/604, 2017. http://eprint.iacr.org/2017/604.
29. A. Al Jabri. *A Statistical Decoding Algorithm for General Linear Block Codes*, pages 1–8. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
30. Ghazal Kachigar and Jean-Pierre Tillich. Quantum information set decoding algorithms. In Tanja Lange and Tsuyoshi Takagi, editors, *PQCrypto 2017*, volume 10346 of *LNCS*, pages 69–89. Springer, 2017.
31. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*, volume 16. North-Holland Mathematical Library, 1977.
32. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report*, 44:114–116, January 1978.
33. R. Misoczki and P. S. L. M. Barreto. Compact mceliece keys from goppa codes. In *Selected Areas in Cryptography*, pages 376–392, 2009.
34. R. Niebuhr. *Statistical Decoding of Codes over $\mathbb{F}_q$*, pages 217–227. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
35. R. Niebuhr, E. Persichetti, P.-L. Cayrel, S. Bulygin, and J. Buchmann. On lower bounds for information set decoding over $\mho_q$ and on the effect of partial knowledge. *Int. J. Inf. Coding Theory*, 4(1):47–78, January 2017.
36. R. Nojima, H. Imai, K. Kobara, and K. Morozov. Semantic security for the McEliece cryptosystem without random oracles. *Des. Codes Cryptography*, 49(1-3):289–305, 2008.
37. Edoardo Persichetti. Compact mceliece keys based on quasi-dyadic srivastava codes. *Journal of Mathematical Cryptology*, 6(2):149–169, 2012.
38. Edoardo Persichetti. Secure and anonymous hybrid encryption from coding theory. In Philippe Gaborit, editor, *Post-Quantum Cryptography: 5th International Workshop, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings*, pages 174–187, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

39. C. Peters. Information-set decoding for linear codes over $\mathbb{F}_q$. In *PQCrypto*, LNCS, pages 81–94, 2010.
40. E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions*, IT-8:S5–S9, 1962.
41. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
42. F. Strenzke. A timing attack against the secret permutation in the mceliece pkc. In *PQCrypto*, pages 95–107, 2010.
43. F. Strenzke, E. Tews, H. G. Molter, R. Overbeck, and A. Shoufan. Side channels in the mceliece pkc. In *PQCrypto*, pages 216–229, 2008.

A Note on the choice of ω

In this section we point out some considerations about the choice of the offset ω during the key generation process.

The usual decoding algorithm for alternant codes, for example as in [31], relies on the special form of the parity-check matrix ($H_{ij} = y_j x_j^{i-1}$). The first step is to recover the error locator polynomial $\sigma(x)$, by means of the euclidean algorithm for polynomial division; then it proceeds by finding the roots of σ. There is a 1-1 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 nontrivial 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 nonzero.

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