Compact McEliece keys based on Quasi-Dyadic Srivastava codes

Edoardo Persichetti

Department of Mathematics, University of Auckland, New Zealand.

e.persichetti@math.auckland.ac.nz

Abstract

The McEliece cryptosystem is one of the few systems to be considered secure against attacks by Quantum computers. The original scheme is built upon Goppa codes and produces very large keys, hence recent research has focused mainly on trying to reduce the public key size. Previous proposals tried to replace the class of Goppa codes with other families of codes, but this was revealed to be an insecure choice. In this paper we introduce a construction based on Generalized Srivastava codes, a large class which includes Goppa codes as a special case, that allows relatively short public keys without being vulnerable to known structural attacks.

1 Introduction

Recent public-key cryptography is largely based on number theory problems, the most popular being factoring and computing discrete logarithms. These systems constitute an excellent choice in many applications, and their level of security is well defined and understood. One of the major drawbacks, though, is that they will be very vulnerable once quantum computers of an appropriate size are available. There is then a strong need for alternative systems that would resist even attackers equipped with quantum technology.

One of the most well-known systems believed to be secure even against quantum attacks is the McEliece cryptosystem. It is based on algebraic coding theory, and has a very fast and efficient encryption procedure. The original McEliece [14], introduced in 1978, uses binary Goppa codes as a basis for the construction. Though this proved to be very resistant against all known attacks, it has one big flaw: the size of the public key. In fact, the public key size for the original parameters ([1024,524]-code with error correction capacity of 50) proposed by McEliece is 67,072 bytes, against the 256 bytes of a 1024-bit Modulus instance of RSA. Thus, during the last years, research has focused mainly on finding a way to significantly reduce the size of the public key. The alternative version provided by Niederreiter, for example, nearly halves (to 32,750 bytes) the public key size, but this is still huge compared to a desirable key size, and clearly impractical on constrained devices.

To have a more compact public key it has been proposed to use codes with particular structures. Two examples are given by Misoczki and Barreto with dyadic matrices [15] and Berger et al. with quasi-cyclic codes [3]. This approach does lead to very small public keys, e.g. 4,096 bits (512 bytes).
Unfortunately, modifying the structure of the codes exposes the cryptosystems to the so-called

structural attacks (algebraic attacks). These attacks aim to exploit the hidden structure, in order to recover the private key. Almost all of the variants presented until now have been broken or proven to be insecure mostly due to an attack presented by Faugère et al. in [9].

Our scheme is based on Generalized Srivastava codes and represents a generalization of [15], with the advantage of a better flexibility. This comes mainly from the fact that Goppa codes are a subclass of Generalized Srivastava codes, corresponding to a particular choice of parameters. In our construction the parameters can instead be chosen in different ways, in order to maximize the reduction in the key size, or to comply with higher levels of security.
In particular, we claim a greater resistance to the known structural attacks, while the keys have similar size to the ones presented in [15].
The paper is organized as follows: in Section 2 we introduce definitions and concepts from coding theory. In Section 3, we give a brief summary of the McEliece cryptosystem, and Goppa codes. Section 4 is the central part of the paper and contains a precise description of the construction. More details about security are given in Section 5, as well as some sample parameters. Finally, we conclude in Section 6.

2 Preliminaries

We present here some definitions we will need to define our scheme.

Definition 1 Given a ring R (in our case the finite field $\mathbb{F}_{q^m}$) and a vector $\bar{h} = (h_0, \ldots, h_{n-1}) \in \mathsf{R}^n$, the *dyadic* matrix $\Delta(\bar{h}) \in \mathsf{R}^{n \times n}$ 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 $\bar{h}$ is called its signature. Moreover, $\Delta(t, \bar{h})$ denotes the matrix $\Delta(\bar{h})$ 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 2 Given two disjoint sequences $\bar{v} = (v_1, \ldots, v_\ell) \in \mathbb{F}_q^\ell$ and $\bar{L} = (L_1, \ldots, L_n) \in \mathbb{F}_q^n$, the *Cauchy matrix* $C(\bar{v}, \bar{L})$ is the matrix with components $C_{ij} = \dfrac{1}{v_i - L_j}$, i.e.

$$C(\bar{v}, \bar{L}) = \begin{pmatrix} \dfrac{1}{v_1 - L_1} & \cdots & \dfrac{1}{v_1 - L_n} \\ \vdots & \vdots & \vdots \\ \dfrac{1}{v_\ell - L_1} & \cdots & \dfrac{1}{v_\ell - L_n} \end{pmatrix}.$$

Cauchy matrices have the property that all of their submatrices are invertible [20].

Definition 3 Given a sequence $\bar{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$, the *Vandermonde matrix* of order ℓ, $V(\ell, \bar{x})$, is the matrix with components $V_{ij} = x_j^{i-1}$, i.e.

$$V(\ell, \bar{x}) = \begin{pmatrix} 1 & \ldots & 1 \\ x_1 & \ldots & x_n \\ \vdots & \vdots & \vdots \\ x_1^{\ell-1} & \ldots & x_n^{\ell-1} \end{pmatrix}.$$

Definition 4 Given two sequences $\bar{x} = (x_1, \ldots, x_n)$, $\bar{y} = (y_1, \ldots, y_n) \in \mathbb{F}_{q^m}^n$, a *Generalized Reed-Solomon (GRS) code* of order ℓ is defined by a parity-check matrix related to the Vandermonde form, more precisely $V(\ell, \bar{x}) \cdot \mathrm{Diag}(\bar{y})$, i.e. the matrix with components $A_{ij} = y_j x_j^{i-1}$:

$$H = \begin{pmatrix} y_1 & \ldots & y_n \\ y_1 x_1 & \ldots & y_n x_n \\ \vdots & \vdots & \vdots \\ y_1 x_1^{\ell-1} & \ldots & y_n x_n^{\ell-1} \end{pmatrix}.$$

If the resulting code is then restricted to $\mathbb{F}_q$ it is called an *Alternant code*.

Definition 5 For $m, n, s, t \in \mathbb{N}$ and a prime power q, let $\bar{\alpha} = (\alpha_1, \ldots, \alpha_n)$, $\bar{w} = (w_1, \ldots, w_s)$ 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

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

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 also part of the Alternant family, as will be shown further in the paper. More information about this class of codes can be found in [13, Ch. 12, §6].

3 The McEliece Cryptosystem

In this section we give the basic notions about the McEliece Cryptosystem, as well as Goppa codes, on which the whole cryptosystem relies.
The general framework proceeds as follows:

Key Generation: Pick a $k \times n$ generator matrix G for a w-error correcting linear code over the finite field $\mathbb{F}_q$, an $n \times n$ permutation matrix P and a $k \times k$ invertible matrix S at random, then compute $G' = SGP$, which is another valid generator matrix. This is the public key. The private key consists of G, S, P, and the parameters n, k, w are public.

Encryption: To encrypt a plaintext $x \in \mathbb{F}_q^k$, compute the corresponding codeword xG' and add some errors at random by using an error vector e of weight at most w, obtaining the ciphertext $y = xG' + e$.

Decryption: Given a ciphertext y, calculate $yP^{-1} = xG'P^{-1} + eP^{-1} = xSGPP^{-1} + eP^{-1} = xSG + eP^{-1}$, and since the weight of eP^{-1} is still less or equal to w, it is enough to apply the decoding algorithm for the code to retrieve xS and consequently x.

Since the published code does not come with any recognizable structure, an attacker that does not know the secret key is faced with the *General Decoding Problem (GDP)*, that is, to find the closest codeword in a linear code C to a given vector, assuming that there is a unique closest codeword. This problem is well known and was proved to be NP-complete. Moreover, GDP is believed to be hard on average, and not just on the worst-case instances.

A version of the McEliece cryptosystem that uses the parity-check matrix instead of the generator matrix has been proposed by Niederreiter [17], and has been proved to be completely equivalent in terms of security.

The original McEliece scheme makes use of binary Goppa codes.

Definition 6 Fix a finite field $\mathbb{F}_q$ and an integer $m > 1$. Choose a polynomial $g(z)$ in $\mathbb{F}_{q^m}[z]$ of degree $t < n/m$ and a sequence of distinct elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_{q^m}$ such that $g(\alpha_i) \neq 0$ for all i. The polynomial $g(z)$ is called the *Goppa polynomial*. The set of words $\bar{c} = (c_1, \ldots, c_n) \in \mathbb{F}_{q^m}^n$ with $\sum_{i=1}^{n} \frac{c_i}{z-\alpha_i} \equiv 0 \pmod{g(z)}$ defines an $[n, n-t]$ linear code over $\mathbb{F}_{q^m}$. The corresponding *Goppa code* is the restriction of this code to $\mathbb{F}_q$, i.e. the set of elements $\bar{c} = (c_1, \ldots, c_n) \in \mathbb{F}_q^n$ which satisfy the above condition.

Alternatively (and usually) a Goppa code is defined by means of its parity-check matrix, which is of the form:

$$H = \begin{pmatrix} \frac{1}{g(\alpha_1)} & \cdots & \frac{1}{g(\alpha_n)} \\ \vdots & \vdots & \vdots \\ \frac{\alpha_1^{t-1}}{g(\alpha_1)} & \cdots & \frac{\alpha_n^{t-1}}{g(\alpha_n)} \end{pmatrix}$$

It is clear then that a Goppa code has dimension $k \geq n - mt$. The minimum distance is $t + 1$, or $2t + 1$ in the special binary case ($q = 2$).

More generally, a Goppa code is a particular case of the class of alternant codes, with $x_i = \alpha_i$, $y_i = 1/g(\alpha_i)$. They form a very large family to choose from, and possess an efficient decoding algorithm, which makes them a natural candidate for encryption purposes.

4 Construction

Our proposal is to use GS codes instead of Goppa codes. Note that GS codes are also alternant codes, hence an efficient decoding algorithm exists. According to Sarwate [19, Cor. 2] the complexity of decoding is $\mathcal{O}(n \log^2 n)$, which is the same as for Goppa codes; thus GS codes are another suitable choice for a McEliece-type cryptosystem.
Now, it is evident that an equivalent parity-check matrix (by a row permutation) for a GS code is given by

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

where each block is

$$\hat{H}_i = \begin{pmatrix} \frac{z_1}{(\alpha_1 - w_1)^i} & \cdots & \frac{z_n}{(\alpha_n - w_1)^i} \\ \frac{z_1}{(\alpha_1 - w_2)^i} & \cdots & \frac{z_n}{(\alpha_n - w_2)^i} \\ \vdots & \vdots & \vdots \\ \frac{z_1}{(\alpha_1 - w_s)^i} & \cdots & \frac{z_n}{(\alpha_n - w_s)^i} \end{pmatrix}.$$

On the other hand, it can be easily proved that every GS code with $t = 1$ is a Goppa code. We know [13, Ch. 12, Pr. 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). Misoczki and Barreto [15, Th. 2] showed that the intersection of the set of matrices in Cauchy form with the set of dyadic matrices is not empty if the code is defined over a field of characteristic 2, and the dyadic signature satisfies

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

In this case the parameters for the Cauchy matrix form are determined as $v_{i+1} = 1/h_i + \omega$ and $L_{j+1} = 1/h_j + 1/h_0 + \omega$ for a certain offset $\omega \in \mathbb{F}_{q^m}$.

We use these facts, and the algorithm given by Barreto et al. in [2, Algorithm 2], which is an improvement to the one presented in [15], to build a GS code in quasi-dyadic form.

1. Fix a finite field $\mathbb{F}_{q^m} = \mathbb{F}_{2^u}$ where $q = 2^\lambda$, $u = m\lambda$. Choose a code length $n < q^m$, $n = n_0 s$, and s being a power of 2. The parameters s, t are chosen such that $mst < n$. More details about the choice of s and t will be given later.

2. Produce a valid dyadic signature $\bar{h}$ over $\mathbb{F}_{q^m}$ using the algorithm of [2]. This consists essentially of two steps:

   - Assign nonzero distinct values at random to h_0 and to every h_j for j a power of 2. The remaining elements are formed using (1) for the appropriate choices of i and j. The resulting signature will have length q^m.
   - Return a selection of blocks of dimension s up to length n, making sure to exclude any block containing an undefined entry.

3. Starting from the signature $\bar{h}$, build the Cauchy support as shown above by choosing at random the offset ω (refer to appendix C about the choice of the offset), then form the first $s \times n$ block in Cauchy fashion. This corresponds to $\hat{H}_1$ by setting $w_i = v_i$, $\alpha_j = L_j$ and remembering that we are in characteristic 2, so that

$$v_i - L_j = v_i + L_j = w_i + \alpha_j = \alpha_j + w_i = \alpha_j - w_i.$$

   Note that this block is dyadic (of order s) as it defines a GS code with $t = 1$, equivalently a Goppa code.

4. Build the remaining blocks by consecutive powers, up to the power of t. This means $\hat{H}_2$ is obtained by squaring each element of $\hat{H}_1$, $\hat{H}_3$ is obtained by cubing, and so on.

5. Pick the z_i uniformly at random with the following restriction:

$$z_{is+j} = z_{is} \text{ for } i = 0, \ldots, n_0 - 1,\ j = 1, \ldots, s.$$

6. The final matrix will be $H = \hat{H} \cdot \mathrm{Diag}(z_i)$. Project H over the base field to obtain an $mst \times n$ parity-check matrix, which can be row-reduced to the systematic form $H^* = (M|I_{n-k})$, having $k = n - mst$ with high probability (see appendix A).
   Note that all of these operations preserve the dyadic structure, since the powering process acts on every single element, the z_i are chosen to be equal s-wise and all the operations occurring during the row reducing are performed block by block in the ring of dyadic matrices over $\mathbb{F}_q$. Hence H^* and in particular M will be still formed by dyadic blocks.

7. The public key is the generator matrix $G = (I_k|M^T)$. Since M^T is $k \times (n-k) = k \times mst$ and is $s \times s$ block dyadic, it requires only $kmst/s = kmt$ field elements for storage, equivalent to $kmt\lambda$ bits.

The public key G just generated can be used, for instance, as a trapdoor for a McEliece scheme, or equivalently H^* can serve as trapdoor for a Niederreiter scheme. In the following section we analyze the security of a scheme based on quasi-dyadic trapdoors.

Some remarks about the construction: the algorithm presented by Barreto et al. runs in polynomial time. Since every element of the signature is assigned a value exactly once, the running time is $\mathcal{O}(q^m)$ $\mathcal{O}(n)$ steps. The authors in [15] did not give a lower bound for the number of possible *distinct* codes, but only the upper bound $\binom{N/t}{\ell} \cdot \ell! \cdot t^\ell \cdot \prod_{i=0}^{\lceil \log N \rceil} (q - 2^i)$ (due to, respectively, selection, rearrangement, permutations of the blocks and number of signatures generated by the algorithm). It is believed that the algorithm does produce close to this number of codes, but it is too hard to actually state the exact number of *distinct* codes constructible.

5 Security

Misoczki and Barreto in [15] give an assessment of the hardness of decoding quasi-dyadic codes, providing a reduction to the Syndrome Decoding problem.
As with the original McEliece cryptosystem, the quasi-dyadic variant is susceptible to general decoding attacks. The best attack known at the moment is considered to be *Information Set Decoding*, which was recently generalized to codes over $\mathbb{F}_q$ [18]; a new version of the attack is presented in [7] under the name of *Ball-Collision Decoding*, but the improvement in the decoding time seems to be relevant only asymptotically and a generalization of ball-collision decoding to the nonbinary case has not yet been presented. Since decoding attacks don't take into account the special properties of the code but just the length, dimension, and number of errors introduced, these parameters (which are all related) need to be carefully chosen.

Recently, a very effective structural attack has been presented by Faugère, Otmani, Perret and Tillich [9]. It relies on the fundamental property $H \cdot G^T = 0$ to build an algebraic system, using then Gröbner bases techniques to solve it. The special properties of Goppa codes in dyadic form are of key importance, as they contribute to considerably reduce the number of unknowns of the system. Also, the extension degree m comes into account as it defines the dimension of the solution space.

Idea of the attack: a $k \times n$ generator matrix $G = \{g_{i,j}\}$ is given as public key, G being a matrix formed of $\ell \times \ell$ blocks, with $k = k_0\ell, n = n_0\ell$. Since every Goppa code is an alternant code, a parity-check matrix will exist in the form $H' = \{y_j x_j^i\}$; these elements are represented by two sets of unknowns $\{X_i\}$ and $\{Y_i\}$. We then obtain the following system of equations:

$$\left\{ g_{i,0} Y_0 X_0^j + \cdots + g_{i,n-1} Y_{n-1} X_{n-1}^j = 0 \mid i = 0, \ldots, k-1, j = 0, \ldots, \ell - 1 \right\}. \tag{2}$$

Some relations are derived [9, Pr. 5] from the dyadicity and from the algorithm used to build the signature, namely:

$$\begin{cases} Y_{j\ell+i} = Y_{j\ell} \\ X_{j\ell+i} + X_{j\ell} = X_i + X_0 \\ X_{j\ell+(i\oplus i')} = X_{j\ell+i} + X_{j\ell+i'} + X_{j\ell} \end{cases} \tag{3}$$

for any $0 \le j \le n_0 - 1$ and $0 \le i, i' \le \ell - 1$.

Lemma 1 ([9]) *The system has (after applying the relations):*

- $n_0 - 1$ *unknowns* Y_i
- $n_0 - m$ *linear equations involving only the* Y_i *(the case* $j = 0$ *in equation (2))*
- $n_0 - 2 + \log_2 \ell$ *unknowns* X_i
- $\ell(\ell - 1)(n_0 - m)$ *nonlinear equations containing monomials of the form* $Y_i X_i^\eta$ *for* $\eta > 0$.

Proof The first property states that the Y_i of each block are all equal, thus there are $n/\ell = n_0$ distinct variables, and we can arbitrarily choose one of them, which explains $n_0 - 1$. Moreover, because of the dyadicity of G, the linear equations in the Y_i are identical, hence redundant, for all the rows of each dyadic block. So we have $k/\ell = (n - m\ell)/\ell = (n_0\ell - m\ell)/\ell = \ell(n_0 - m)/\ell = n_0 - m$ linear equations as claimed.
The other two are a direct consequence of the second and third properties: in fact, we can fix arbitrarily two variables, say X_0 and X_ℓ and express every other in terms of those two for each block, which means $n_0 + \log_2 \ell - 2$. △

The attacker first tries to deduce a simpler system involving only the X_i: fix the free Y_i variables and rewrite the remaining as a function of those, then substitute into the equations. If it is possible to find the free variables (if the number of those is very small, even just by guessing) the computation of the desired simplified system follows immediately. Since there are $n_0 - 1$ variables and $n_0 - m$ equations, we have exactly $n_0 - 1 - (n_0 - m) = m - 1$ free variables; thus it is important for security to keep the extension degree m high (the authors in [10] indicate that this number should be not smaller than 20). Once the Y_i are removed, the second step is to linearize the system by using the following trick: observe that each block of equations is now homogeneous, then consider only those whose degree corresponds to a power of 2, and discard the rest. There are exactly $\lfloor \log_2(\ell - 1) \rfloor = \log_2(\ell - 1)$ such equations, which usually allows to recover the X_i. A more detailed description can be found in [9].

It is clear that, since GS codes also belong to the class of alternant codes, this framework can be applied to our proposal; as we will see, all the properties hold in a similar way.
While a detailed complexity analysis has not yet been given, it makes sense to compare the two security levels. In fact, we can think of a Goppa code or a GS code with the same parameters $[n, k, d]$ having, respectively, $k = n - m\ell = n - mst \Longrightarrow \ell = st$.
If $t = 1$ then our scheme is exactly the same as [15]. For $t > 1$, however, the system parameters change, as $n = n_0\ell = n_0' s$ having $n_0' > n_0$. We now focus our attention on the linear part: just like before, it is possible to prove that all the Y_i in a block are equal.

Proposition 1 *Let Y_i be the set of unknowns defined in (2). Then:*

$$Y_{is+j} = Y_{is} \text{ for } i = 0, \ldots, n_0 - 1, j = 1, \ldots, s.$$

Proof Recall from Definition 4 the generic form for a parity-check matrix in alternant form:

$$H = \begin{pmatrix} y_1 & \cdots & y_n \\ y_1 x_1 & \cdots & y_n x_n \\ \vdots & \vdots & \vdots \\ y_1 x_1^{\ell-1} & \cdots & y_n x_n^{\ell-1} \end{pmatrix}.$$

However, the general alternant form for Srivastava codes is given in a slightly different way. In fact, if we consider an invertible $\ell \times \ell$ matrix C, then $H' = CH$ is also a parity-check matrix:

$$H = \begin{pmatrix} c_{1,1} & \cdots & c_{1,\ell} \\ c_{2,1} & \cdots & c_{2,\ell} \\ \vdots & \vdots & \vdots \\ c_{\ell,1} & \cdots & c_{\ell,\ell} \end{pmatrix} \begin{pmatrix} y_1 & \cdots & y_n \\ y_1 x_1 & \cdots & y_n x_n \\ \vdots & \vdots & \vdots \\ y_1 x_1^{\ell-1} & \cdots & y_n x_n^{\ell-1} \end{pmatrix} = \begin{pmatrix} y_1 g_1(x_1) & \cdots & y_n g_1(x_n) \\ y_1 g_2(x_1) & \cdots & y_n g_2(x_n) \\ \vdots & \vdots & \vdots \\ y_1 g_\ell(x_1) & \cdots & y_n g_\ell(x_n) \end{pmatrix}$$

where $g_i(x) = c_{i,1} + c_{i,2}x + c_{i,3}x^2 + \cdots + c_{i,\ell}x^{\ell-1}$ for each $i = 1, \ldots, \ell$.

In the case of Srivastava codes we have $\ell = st$, and we set

- $g_{(i-1)t+k}(x) = \dfrac{\prod_{j=1}^{s}(x - w_j)^t}{(x - w_i)^k}$ for $i = 1, \ldots, s$ and $k = 1, \ldots, t$.
- $y_i = \dfrac{z_i}{\prod_{j=1}^{s}(\alpha_i - w_j)^t}$.

It is easy to see that with these settings, we get the parity-check matrix given in Definition 5.
Now, we want to prove that $y_{is+j} = y_{is}$ for $i = 0, \ldots, n_0 - 1, j = 1, \ldots, s$.
Let's then fix a specific i (i.e. choose a block) and consider in particular $y_{is+j^*} = y_{is}$, for any $j^* \in \{1, \ldots, s\}$.

If we can prove that $\prod_{j=1}^{s} (\alpha_{is+j^*} - w_j) = \prod_{j=1}^{s} (\alpha_{is} - w_j)$, then obviously

$$\prod_{j=1}^{s} (\alpha_{is+j^*} - w_j)^t = \prod_{j=1}^{s} (\alpha_{is} - w_j)^t \Longrightarrow \frac{1}{\prod_{j=1}^{s} (\alpha_{is+j^*} - w_j)^t} = \frac{1}{\prod_{j=1}^{s} (\alpha_{is} - w_j)^t}.$$

We know that $z_{is+j} = z_{is}$ for $i = 0, \ldots, n_0 - 1, j = 1, \ldots, s$ by construction.

Hence $\dfrac{z_{is+j}}{\prod_{j=1}^{s} (\alpha_{is+j^*} - w_j)^t} = \dfrac{z_{is}}{\prod_{j=1}^{s} (\alpha_{is} - w_j)^t}$, and this means $y_{is+j} = y_{is}$.

Since this does not depend on the choice of i, it is then true for all i, and we obtain our result.

It remains to prove $\prod_{j=1}^{s} (\alpha_{is+j^*} - w_j) = \prod_{j=1}^{s} (\alpha_{is} - w_j)$.

Now, remember that, by means of the algorithm, the support was built as $w_{i+1} = v_{i+1} = 1/h_i + \omega$ and $\alpha_{j+1} = L_{j+1} = 1/h_j + 1/h_0 + \omega$, so our expression becomes

$$\prod_{j=1}^{s} (1/h_{is+j^*-1} + 1/h_0 - 1/h_{j-1}) = \prod_{j=1}^{s} (1/h_{is-1} + 1/h_0 - 1/h_{j-1})$$

or, without loss of generality, rearranging and noting that we are in characteristic 2,

$$\prod_{j=1}^{s} (1/h_0 + 1/h_{is+j^*} + 1/h_j) = \prod_{j=1}^{s} (1/h_0 + 1/h_{is} + 1/h_j).$$

Let $k_1 = is + j^*$ and $k_2 = is$; then, remembering equation (1), we can rewrite:

$$\prod_{j=1}^{s} (1/h_0 + 1/h_{k_1} + 1/h_j) = \prod_{j=1}^{s} (1/h_0 + 1/h_{k_2} + 1/h_j) \Longleftrightarrow \prod_{j=1}^{s} (1/h_{k_1 \oplus j}) = \prod_{j=1}^{s} (1/h_{k_2 \oplus j})$$

$$\Longleftrightarrow \frac{1}{\prod_{j=1}^{s} h_{k_1 \oplus j}} = \frac{1}{\prod_{j=1}^{s} h_{k_2 \oplus j}} \Longleftrightarrow \prod_{j=1}^{s} h_{k_1 \oplus j} = \prod_{j=1}^{s} h_{k_2 \oplus j},$$

which is true since k_1 and k_2 belong to the same block (the matrix is $s \times s$ dyadic).
Essentially, this corresponds to multiplying together the elements of a string of length s (substring of a row) on two different rows of the same block, which, because of the dyadicity, we know being just a rearrangement one of the other. Hence the equality holds, and this terminates the proof. $\triangle$

Proposition 1 tells us that there are $n'_0 - 1$ distinct variables (like before, we can arbitrarily fix one of them). Now, the dimension of the blocks is smaller (as $s < \ell$), so we will have more equations, but the numbers are not increasing at the same rate. In fact $k/s = (n - mst)/s = (n'_0 s - mst)/s = s(n'_0 - mt)/s = n'_0 - mt$. We will then have:

- $n'_0 - 1$ unknowns Y_i
- $n'_0 - mt$ linear equations

which give a solution space of dimension $mt - 1$. This is a major improvement since now the security does not rely entirely and only on m; we can instead increase t so that we are not forced to use a big extension field, which gives large and unpractical keys, while making the attack less effective.

In the following tables we give various sets of parameters in order to better illustrate the features of our scheme. We also include experimental results about resistance to the attack just presented (column "FOPT cost"); these are obtained by using the upper bound provided by equation (13) in [10, Section 6]. We remark that the resulting numbers are just a theoretical upper bound that gives the approximate cost of computing a Gröbner basis with the indicated dimensions and variables, but nevertheless are useful to give an idea of the expected cost of the attack against that specific set of parameters. The numbers obtained by the theorem match with the costs obtained for the attacks successfully mounted against the codes of [3] and [15]. It also seems to emerge why the authors indicate 20 as a safe threshold, since all the parameters that produce a number of free variables greater than 20 generate a complexity superior to 2^{128}.

Table 1 highlights the differences in performance and security according to the choice of m and t when keeping fixed the other parameters. The column "ISD cost" refers to the estimated complexity of decoding attacks[1]($\log_2$ of binary operations). Note that the first line ($t = 1$) represents a Goppa code.

Table 1: Example of parameters for GS codes over the base field $\mathbb{F}_{2^2}$, for a fixed number ($mt - 1 = 23$) of free variables. The column "Size" refers to the public key size.

m	n	k	s	t	Errors	Size (bytes)	ISD cost	FOPT cost
24	12288	6144	2^8	1	128	36864	128	150
12	6144	3072	2^7	2	128	18432	128	150
8	4096	2560	2^6	3	96	15360	128	160

Here we chose to keep constant this particular number of free variables mainly because $mt = 24$ gives a lot of possibilities for factoring (i.e. a lot of different choices for m and t) and the resulting amount 23 is well above the threshold of 20 indicated in [10].
It is also possible to observe that choosing an odd value for t gives better results even with a smaller number of errors introduced (e.g. compare line 2 and 3). That is because while the product st decreases (and consequently the numbers of correctable errors), so do the code minimum requirements for size (n) and dimension (k). This allows a tighter choice of parameters and overall works better for our purposes.

[1]To compute this number we refer to [18] and use the corresponding script provided by Christiane Peters in http://www2.mat.dtu.dk/people/C.Peters/isdfq.html.

Table 2: GS codes over the base field $\mathbb{F}_{2^2}$ with fixed length $n = 1920$ and extension degree $m = 6$.

k	s	t	Errors	Size (bytes)	ISD cost	FOPT cost
960	2^5	5	80	7200	90	186
768	2^6	3	96	3456	80	105

From Table 2 it is evident that a bigger t allows the construction of a code with better performance, but results in a much bigger key. It is also clear how deeply all the parameters are intertwined, at the same time contributing to the flexibility of the scheme: the first code, for instance, generates a much greater complexity against the structural attack, while achieving an even smaller key size than any of the codes in Table 1. However, the security against general decoding attacks decreases considerably.
Keeping all of this in mind, we give in Table 3 a sample of some smaller codes with the aim to minimize the public key size.

Table 3: Sets of parameters for smaller GS codes, obtained by choosing larger base fields and increasing t, while lowering the extension degree.

Base Field	m	n	k	s	t	Errors	Size (bytes)	ISD cost	FOPT cost
$\mathbb{F}_{2^5}$	2	992	416	2^5	9	144	4680	128	105
$\mathbb{F}_{2^4}$	3	768	432	2^4	7	56	4536	80	132
$\mathbb{F}_{2^5}$	2	512	256	2^4	2^3	64	2560	80	96

6 Conclusions

We have given a detailed description of a construction based on Quasi-Dyadic Generalized Srivastava codes. This is a generalization of [15], and is suitable as a trapdoor for a McEliece or Niederreiter scheme. The public keys are considerably smaller than the original McEliece's proposal, and the construction easily gives codes secure against general decoding attacks.

Thanks to the introduction of the parameter t we are able to modulate our scheme in a much more flexible way, allowing us to consider codes over smaller extension fields without losing in security; moreover, the parameter t balances both the ratio (extension degree)/(number of free variables), and the reduction in the public key size, as this depends solely on s, which grows or shrinks according to t (for a fixed dimension and error-correction capacity).
The result of this is a flexible and practical scheme which produces very small keys and resists all the attacks presented so far. As a comparison, take the codes presented in [15]:

Table 4: Quasi-Dyadic Goppa codes ([15, Table 2]) over $\mathbb{F}_2$ and with extension degree $m = 16$.

n	k	ℓ	Size (bytes)	ISD cost	FOPT cost
8192	4096	256	8192	256	99
6912	2816	256	5632	192	95
4092	2048	128	4096	128	98
3584	1536	128	3072	112	95
2304	1280	64	2560	80	100

For all these codes, the level of security ($m - 1 = 15$) against the FOPT attack [9] is the same of the last code in Table 3, but only one has the same key size (2560 bytes), whereas the others are all considerably larger. If our main concern is resistance against structural attacks rather than general decoding attacks, it is then evident that we have an advantage.
An example could be represented by the codes in Table 3, line 1 and Table 4, line 3. For the same security level of 2^{128} we have a solution space of dimension $mt - 1 = 17$ for the former as opposed to 15 for the latter.

We remark that until a precise complexity analysis for the structural attacks is given, we should obey the condition obtained from the experimental results presented in [10], thus keeping the dimension of the solution space for the Y_i strictly greater than 20.

The choice of a base field other than $\mathbb{F}_2$, though actually increasing the public key size, looks like a better choice for the construction. Unlike the case of Goppa codes, GS codes do not benefit from an increased error-correction capacity in the binary case, so there is no particular reason to choose binary over nonbinary. Instead, choosing a bigger base field allows us to further reduce the extension degree to values for which the scheme would otherwise be infeasible.
We have also given (appendix A) an estimate of the probability of getting a full-rank matrix after the projection on the base field. This is necessary to be sure that the key generation algorithm is efficient. Since also this probability depends on the choice of the base field, this is yet another reason to choose nonbinary codes.

Further ideas of research include developing a precise security analysis which would allow a better optimization of the parameters. An implementation of a McEliece encryption scheme using quasi-dyadic GS codes is currently being designed; it would be interesting to apply the same framework to other cryptographic applications such as, for example, signature schemes (as in [2]).

References

[1] A. A. de Andrade and R. Palazzo Jr. "Goppa and Srivastava codes over finite rings." In *Comp. Appl. Math*, volume 24, issue 2, pages 231-244, 2005.

[2] P. S. L. M. Barreto, P.-L. Cayrel, R. Misoczki, and R. Niebuhr, "Quasi-dyadic CFS signatures". In *Inscrypt 2010*, LNCS, Springer, October 2010.

[3] T. P. Berger, P. L. Cayrel, P. Gaborit and A. Otmani. "Reducing key length of the McEliece cryptosystem". In Bart Preneel, editor, *Progress in Cryptology - Second International Conference on Cryptology in Africa (AFRICACRYPT 2009)*, volume 5580 of *Lecture Notes in Computer Science*, pages 77-97, Gammarth, Tunisia, June 21-25, 2009.

[4] T. P. Berger and P. Loidreau. "How to mask the structure of codes for a cryptographic use". In *Design, Codes and Cryptography*, volume 35, pages 63-79, 2005.

[5] T. P. Berger and P. Loidreau. "Designing an efficient and secure public-key cryptosystem based on reducible rank codes". In *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 218-229, 2004.

[6] T. P. Bernstein, T. Lange and C. Peters. "Attacking and defending the McEliece cryptosystem". In J. Buchman and J. Ding, editors, *Post-Quantum Cryptography- Second International Workshop (PQCrypto 2008)*, volume 5299 of *Lecture Notes in Computer Science*, pages 31-46, Springer, Berlin, 2008.

[7] T. P. Bernstein, T. Lange and C. Peters, "Smaller Decoding Exponents: Ball-Collision Decoding". In *Advances in Cryptology - CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 743-760, Santa Barbara, CA, USA, August 14-18, 2011.

[8] T. P. Bernstein, T. Lange, C. Peters and H. C. A. van Tilborg, "Explicit bounds for generic decoding algorithms for code-based cryptography". In *Pre-proceedings of WCC 2009*, pages 168-180, 2009.

[9] 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, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 279-298, French Riviera, May 30 - June 3, 2010.

[10] J. C. Faugère, A. Otmani, L. Perret and J. P. Tillich, "Algebraic Cryptanalysis of Compact McEliece's Variants - Toward a Complexity Analysis". preprint 2010.

[11] P. J. Lee and E. F. Brickell, "An observation of the security of McEliece's public-key cryptosystem". In G. Günther, editor *Advances in cryptology - EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 275-280, Springer, Berlin, 1988.

[12] J. S. Leon, "A probabilistic algorithm for computing minimum weights of large error-correcting codes". In *IEEE Transactions on Information Theory*, volume 34, issue 5, pages 1354-1359, 1988.

[13] F. J. MacWilliams and N. J. Sloane, "The theory of error-correcting codes". North Holland, Amsterdam, 1977.

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

[15] R. Misoczki and P. S. L. M. Barreto, "Compact McEliece keys from Goppa codes". In *Selected Areas in Cryptography (SAC 2009)*, Calgary, Canada, August 13-14, 2009.

[16] R. Niebuhr and J. Buchmann, "Application of Algebraic-Geometric Codes in Cryptography", Master's Thesis, 2006.

[17] H. Niederreiter, "A public-key cryptosystem based on shift register sequences". In *EUROCRYPT*, volume 219 of *Lecture Notes in Computer Science*, pages 35-39, 1985.

[18] C. Peters, "Information-set decoding for linear codes over $\mathbb{F}_q$". In *Post-Quantum Cryptography, Third International Workshop, (PQCrypto 2010)*, volume 6061 of *Lecture Notes in Computer Science*, pages 81-94, Darmstadt, Germany, May 25-28, 2010.

[19] D. V. Sarwate, "On the complexity of decoding Goppa codes". In *IEEE Transactions on Information Theory*, volume 23, issue 4, pages 515-516, 1977.

[20] S. Schechter, "On the inversion of certain matrices". In *Mathematical Tables and Other Aids to Computation*, volume 13, issue 66, pages 73-77, 1959.

[21] J. R. Silvester, "Determinants of block matrices". In *The Matematical Gazette*, volume 84, pages 460-467, 2000.

[22] J. Stern, "A method for finding codewords of small weight". In G. D. Cohen and J. Wolfmann, editors, *Coding theory and applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106-113, Springer, New York, 1989.

A Full-rank matrices

We give an estimate of the expected probability of having an invertible submatrix after the co-trace operation defined in point 6 of the key generation algorithm, so that row reduction to the systematic form is actually possible. To do this, we start by considering random matrices as a general case.

Lemma 2 *Let M be a random $n \times n$ matrix over the finite field $\mathbb{F}_q$. Then the probability that M is nonsingular is:*

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

Proof A matrix M is nonsingular if and only if its rows are linearly independent vectors. The choices for the first row are $q^n - 1$, while for each row after the first, we have to be sure that it is not in the span of the previous vectors; hence for the i-th row we have only $q^n - q^{i-1}$ choices. This gives $(q^n - 1)(q^n - q) \dots (q^n - q^{n-1})$ choices over the total q^{n^2}, which is what we wanted to prove. △

Now, we take into account the special form of our matrix. Since it is dyadic, the number of choices for the row vectors is restricted, since every time we choose a row, the following $s - 1$ are uniquely determined according to the dyadic form (permutations). Practically speaking, we are considering an $r \times r$ quasi-dyadic matrix, where $r = mst = r_0 s$, and we are choosing only r_0 row vectors.

However now, in each choice, we must also ensure that the set of s rows produced is by itself linearly independent. Since each of those is composed by r_0 square blocks of side s, we first focus on a single block.

Lemma 3 *Let $D = \Delta(\bar{h})$ be an $s \times s$ matrix over the finite field $\mathbb{F}_{q^m}$ ($q = 2^\lambda$) given by the signature $\bar{h} = (h_0, \dots, h_{s-1})$, with s being a power of 2. Then:*

$$D \text{ is singular} \iff \sum_{i=0}^{s-1} h_i = 0.$$

Proof Since s is a power of 2, say 2^j, we know D is of the following form:

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

where A, B are dyadic submatrices of dimension 2^{j-1} defined, respectively, by $\bar{h}_A = h_0, \dots, h_{s/2-1}$ and $\bar{h}_B = h_{s/2}, \dots, h_{s-1}$. All we need is to consider the determinant of D that, thanks to an easy generalization of [21], we can claim (see appendix B) is equal to $\det(A^2 + B^2)$.
Applying the argument recursively (and remembering that we are in characteristic 2) we arrive at the conclusion that $\det D = (h_0 + \dots + h_{s-1})^{2^j}$. Now, D is singular $\iff \det D = 0 \iff (h_0 + \dots + h_{s-1})^{2^j} = 0 \iff h_0 + \dots + h_{s-1} = 0$, which terminates the proof. △

Thanks to Lemma 3 it is now easy to give a description of how to select the first row. We call a row vector v *good* if the set of s vectors consisting of v and its dyadic rearrangements is linearly independent, and we call v *bad* if it is not good. Now, for every choice of $s-1$ field elements, the sum will still be a field element; hence, for each block we have q^{s-1} signatures that sum to 0, and overall $(q^{s-1})^{r_0}$ bad sequences. It is then sufficient to subtract this number from the total possible choices q^r, and we obtain that the number of good choices for the first row vector is:

$$q^r - (q^{s-1})^{r_0} = q^r - q^{r_0(s-1)} = q^r - q^{r-r_0} = q^{r-r_0}(q^{r_0} - 1).$$

Let's call $\mathcal{G}$ the set of all good rows. As a last precaution, we need to determine how many linear combinations of the rows in a size-s set produce a row which is still in $\mathcal{G}$, so that we can exclude them at the moment of choosing the next one.

This is easy for the first set.

Lemma 4 *Let $v^{(1)}, \ldots, v^{(s)}$ be the first s row vectors of a quasi-dyadic matrix, and suppose the first row is good. Then for every $v = \sum_{i=1}^{s} a_i v^{(i)}$:*

$$v \in \mathcal{G} \iff \sum_{i=1}^{s} a_i \neq 0.$$

Proof Let's analyze, without loss of generality, the first block and write $v_1 + v_2 + \cdots + v_s = (a_1 v_1^{(1)} + a_2 v_1^{(2)} + \cdots + a_s v_1^{(s)}) + (a_1 v_2^{(1)} + a_2 v_2^{(2)} + \cdots + a_s v_2^{(s)}) + \cdots + (a_1 v_s^{(1)} + a_2 v_s^{(2)} + \cdots + a_s v_s^{(s)}) = (a_1 v_1^{(1)} + a_1 v_2^{(1)} + \cdots + a_1 v_s^{(1)}) + (a_2 v_1^{(2)} + a_2 v_2^{(2)} + \cdots + a_2 v_s^{(2)}) + \cdots + (a_s v_1^{(s)} + a_s v_2^{(s)} + \cdots + a_s v_s^{(s)}) = a_1 \sum_{i=1}^{s} v_i^{(1)} + a_2 \sum_{i=1}^{s} v_i^{(2)} + \cdots + a_s \sum_{i=1}^{s} v_i^{(s)}$.

Now, each of these sums is exactly the sum of the elements of each row, which because of the dyadicity is constant, say equal to α, and by hypothesis different from 0; hence we can write $\alpha(a_1 + \cdots + a_s) = 0 \iff a_1 + \cdots + a_s = 0$, which terminates the proof. △

According to Lemma 4 then, $q^{s-1}(q-1)$ linear combinations of the rows in the first set produce a row in $\mathcal{G}$. Unfortunately the same reasoning doesn't work when we consider the next sets, as the rows in the next set will sum in principle to a different element (say β, γ etc.). Hence, we can just obtain a lower bound, by excluding all the q^s linear combinations. However, it is reasonable to think that very few linear combinations produce a bad row, so our lower bound is not far from the real value.

Theorem 1 *Let H be an $r \times n$ parity-check matrix over $\mathbb{F}_q$ as in step 6, with $r = mst = r_0 s$. Then the row-reduction to the systematic form for H succeeds with probability at least:*

$$p = \prod_{i=0}^{r_0-1} \left(1 - \frac{1}{q^{r_0}} - \frac{1}{q^{(r_0-i)s}}\right).$$

Proof Follows directly from our last considerations: we get $p = \dfrac{\prod_{i=0}^{r_0-1} \left(q^r - q^{r-r_0} - q^{is}\right)}{q^{r_0 r}}$.
This is a product of r_0 terms and since $q^{r_0 r} = (q^r)^{r_0}$ we can divide each term by q^r and obtain the conclusion. △

Experimental results suggest this number looks roughly like $(q-1)/q$.

B Determinant of block matrices

We state the following result, which we will need to prove Lemma 3:

Lemma 5 *Let D be an $n \times n$ block-symmetric matrix over a finite field F of characteristic 2, i.e. D is in the form:*

$$D = \begin{pmatrix} A & B \\ B & A \end{pmatrix}.$$

where A and B are themselves block-symmetric matrices of dimension $n/2$.
Then $\det D = \det(A^2 + B^2)$.

Proof We know from [21] that $\det \begin{pmatrix} A & B \\ \mathbf{0} & C \end{pmatrix} = \det \begin{pmatrix} A & \mathbf{0} \\ B & C \end{pmatrix} = \det A \det C$.

Now, consider the following product $M = \begin{pmatrix} A & B \\ B & A \end{pmatrix} \begin{pmatrix} A & \mathbf{0} \\ B & \mathbf{I} \end{pmatrix}$. Since A and B are both

symmetric, $A = A^T$, $B = B^T$ and $AB = (AB)^T$, hence we can rewrite the product as:

$$M = \begin{pmatrix} A & B \\ B^T & A \end{pmatrix} \begin{pmatrix} A^T & \mathbf{0} \\ B & \mathbf{I} \end{pmatrix} = \begin{pmatrix} A^2 + B^2 & B \\ B^T A^T + AB & A \end{pmatrix} = \begin{pmatrix} A^2 + B^2 & B \\ (AB)^T + AB & A \end{pmatrix} =$$

$$= \begin{pmatrix} A^2 + B^2 & B \\ \mathbf{0} & A \end{pmatrix}.$$

Looking at determinants, and applying the hypothesis, we read:

$$\det M = \det D \det A = \det(A^2 + B^2) \det A$$

which implies in particular $(\det D + \det(A^2 + B^2)) \det A = 0$ and the result follows immediately if we assume $\det A \neq 0$. However, we don't even need this assumption if we use the following trick: instead of working over F, let's do our calculations over the corresponding polynomial ring $\mathsf{F}[x]$ by defining $A_x = A + x\mathbf{I}$ and $D_x = \begin{pmatrix} A & B \\ B & A_x \end{pmatrix}$.

We obtain $(\det D_x + \det(AA_x + B^2)) \det A_x = 0$ but now this time we are considering a product of polynomials and $\det A_x = \det(A + x\mathbf{I})$ is certainly not the zero polynomial, hence the left-hand side must be.
Thus $\det D_x = \det(AA_x + B^2)$ follows, from which it is enough to put $x = 0$ to get our result. $\triangle$

C 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 [13], 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.