# An efficient attack of a McEliece cryptosystem variant based on convolutional codes

Grégory Landais and Jean-Pierre Tillich

SECRET Project - INRIA Rocquencourt
Domaine de Voluceau, B.P. 105 78153 Le Chesnay Cedex - France
gregory.landais@inria.fr, jean-pierre.tillich@inria.fr

**Abstract.** Löndahl and Johansson proposed last year a variant of the McEliece cryptosystem which replaces Goppa codes by convolutional codes. This modification is supposed to make structural attacks more difficult since the public generator matrix of this scheme contains large parts which are generated completely at random. They proposed two schemes of this kind, one of them consists in taking a Goppa code and extending it by adding a generator matrix of a time varying convolutional code. We show here that this scheme can be successfully attacked by looking for low-weight codewords in the public code of this scheme and using it to unravel the convolutional part. It remains to break the Goppa part of this scheme which can be done in less than a day of computation in the case at hand.

**Keywords**. Code-based cryptography, McEliece cryptosystem, convolutional codes, cryptanalysis.

## 1 Introduction

In [Sho97], Peter Shor showed that all cryptosystems based on the hardness of factoring or taking a discrete logarithm can be attacked in polynomial time with a quantum computer (see [BBD09] for an extensive report). This threatens most if not all public-key cryptosystems deployed in practice, such as RSA [RSA78] or DSA [Kra91]. Cryptography based on the difficulty of decoding a linear code, on the other hand, is believed to resist quantum attacks and is therefore considered as a viable replacement for those schemes in future applications. Yet, independently of their so-called "post-quantum" nature, code-based cryptosystems offer other benefits even for present-day applications due to their excellent algorithmic efficiency, which is up to several orders of complexity better than traditional schemes.

The first code-based cryptosystem is the McEliece cryptosystem [McE78], originally proposed using Goppa codes. Afterwards several code families have been suggested to replace the Goppa codes in this scheme: generalized Reed–Solomon codes (GRS) [Nie86] or subcodes of them [BL05], Reed–Muller codes [Sid94], algebraic geometry codes [JM96], LDPC codes [BBC08], MDPC codes [MTSB12] or more recently convolutional codes [LJ12]. Some of these schemes allow to reduce the public key size compared to the original McEliece cryptosystem while keeping the same level of security against generic decoding algorithms.

However, for several of the aforementioned schemes it has been shown that a description of the underlying code suitable for decoding can be obtained- this breaks the corresponding scheme. This has been achieved for generalized Reed-Solomon codes in [SS92] and for subcodes of generalized Reed-Solomon codes in [Wie10]. In this case, the attack takes polynomial time and recovers the complete structure of the underlying generalized Reed–Solomon code from the public key $G'$. The Reed-Muller code scheme has also been attacked, but this time the algorithm recovering the secret description of the permuted Reed-Muller code has sub-exponential complexity [MS07] which is enough for attacking the scheme with the parameters proposed in [Sid94] but which is not sufficient to break the scheme completely. Algebraic geometry codes are broken in polynomial time but only for low genus hyperelliptic curves [FM08]. Finally, it should be mentioned that a first version of the scheme based on LDPC codes proposed in [BC07] has been successfully attacked in [OTD10] (but the new scheme proposed in [BBC08] seems to be immune to this kind of attack)

and that a variant [BBC$^+$11] of the generalized Reed-Solomon scheme which was supposed to resist to the attack of [SS92] has recently been broken in [GOT12,CGG$^+$12] by an approach which is related to the distinguisher of Goppa codes which is proposed in [**?**,FGO$^+$11].

The original McEliece cryptosystem with Goppa codes is still unbroken. It was modified in [BCGO09,MB09] by considering quasi-cyclic or quasi-dyadic versions of Goppa codes (or more generally of alternant codes in [BCGO09]) in order to reduce significantly the key size. However, in this case it was shown that the added structure allows a drastic reduction of the number of unknowns in algebraic attacks and most of the schemes proposed in [BCGO09,MB09] were broken by this approach. This kind of attack has exponential complexity and it can be thwarted by choosing smaller cyclic or dyadic blocks in this approach in order to increase the number of unknowns of the algebraic system. When the rate of the Goppa code is close to 1 (as is the case in signature schemes for instance [CFS01]) then it has been shown in [FGO$^+$] that the public key can be distinguished from a random public key. This invalidates all existing security proofs of the McEliece cryptosystem when the code rate is close to 1 since they all rely on the hardness of two problems: the hardness of decoding a generic linear code on one hand and the indistinguishability of the Goppa code family on the other hand.

These algebraic attacks motivate the research of alternatives to Goppa codes in the McEliece cryptosystem and it raises the issue of what kind of codes can be chosen in the McEliece cryptosystem. The proposal with convolutional codes made in [LJ12] falls into this thread of research. What makes this new scheme interesting is the fact that its secret generator matrix contains large parts which are generated completely at random and has no algebraic structure as in other schemes such as generalized Reed-Solomon codes, algebraic geometry codes, Goppa codes or Reed-Muller codes.

In [LJ12] two schemes are given. The first one simply considers as the secret key the generator matrix of a time varying tail-biting convolutional code. A scheme for which it is supposed to resist to attacks of time complexity of about $2^{80}$ elementary operations is suggested and has reasonable decoding complexity. This construction presents however the drawback that the complexity of decoding scales exponentially with the security level measured in bits. The authors give a second scheme which is scalable and which is built upon a Goppa code and extends it by adding a generator matrix of a time varying convolutional code.

We study the security of this second scheme in this article. It was advocated that the convolutional structure of the code can not be recovered due to the fact that the dual code has large enough minimum distance. However, we show here that this scheme can be successfully attacked by looking for low-weight codewords in the public code of this scheme. By a suitable filtering procedure of these low weight codewords we can unravel the convolutional part.

The main point which makes this attack feasible is the following phenomenon : the public code of this scheme contains subcodes of much smaller support but whose rate is not much smaller than the rate of the public code. The support of such codes can be easily found by low weight codewords algorithms. It is worthwhile to notice that the code-based KKS signature scheme [KKS97] could be broken with exactly the same approach [OT11]. It turns out that the support of these subcodes reveals the convolutional structure. By suitably puncturing the public code, there is only the Goppa part which remains. Deciphering an encrypted message can then be done because for the concrete parameters example provided in [LJ12], algorithms for decoding general linear codes can be used in this case to decode the Goppa code successfully. This attack works successfully on the parameters proposed in [LJ12] and needs only a few hours of computation. It should be possible to change the parameters of the scheme to avoid this kind of attack. In order to do so an improved attack is suggested in Subsection 5.1, its complexity is analyzed in Section 5. This suggests that it should be possible to repair the scheme by fixing the parameters in a more conservative way. Some indications about how to perform such a task are given in Subsection 5.3.

## 2 The McEliece scheme based on convolutional codes

The scheme can be summarized as follows.

**Secret key.**

- $\boldsymbol{G}_{\text{sec}}$ is a $k \times n$ generator matrix which has a block form specified in Figure 1;
- $\boldsymbol{P}$ is an $n \times n$ permutation matrix;
- $\boldsymbol{S}$ is a $k \times k$ random invertible matrix over $\mathbb{F}_2$.

**Public key.** $\boldsymbol{G}_{\text{pub}} \stackrel{\text{def}}{=} \boldsymbol{S}\boldsymbol{G}_{\text{sec}}\boldsymbol{P}$.

**Encryption.** The ciphertext $\boldsymbol{c} \in \mathbb{F}_2^n$ of a plaintext $\boldsymbol{m} \in \mathbb{F}_2^k$ is obtained by drawing at random $\boldsymbol{e}$ in $\mathbb{F}_2^n$ of weight equal to some quantity $t$ and computing $\boldsymbol{c} \stackrel{\text{def}}{=} \boldsymbol{m}\boldsymbol{G}_{\text{pub}} + \boldsymbol{e}$.

**Decryption.** It consists in performing the following steps

1. Calculating $\boldsymbol{c}' \stackrel{\text{def}}{=} \boldsymbol{c}\boldsymbol{P}^{-1} = \boldsymbol{m}\boldsymbol{S}\boldsymbol{G}_{\text{sec}} + \boldsymbol{e}\boldsymbol{P}^{-1}$ and using the decoding algorithm of the code with generator matrix $\boldsymbol{G}_{\text{sec}}$ to recover $\boldsymbol{m}\boldsymbol{S}$ from the knowledge of $\boldsymbol{c}'$;
2. Multiplying the result of the decoding by $\boldsymbol{S}^{-1}$ to recover $\boldsymbol{m}$.

The point of the whole construction is that if $t$ is well chosen, then with high probability the Goppa code part can be decoded, and this allows a sequential decoder of the time varying convolutional code to decode the remaining errors. From now on we will denote by $\mathscr{C}_{\text{pub}}$ the code with generator matrix $\boldsymbol{G}_{\text{pub}}$ and by $\mathscr{C}_{\text{sec}}$ the code with generator matrix $\boldsymbol{G}_{\text{sec}}$.

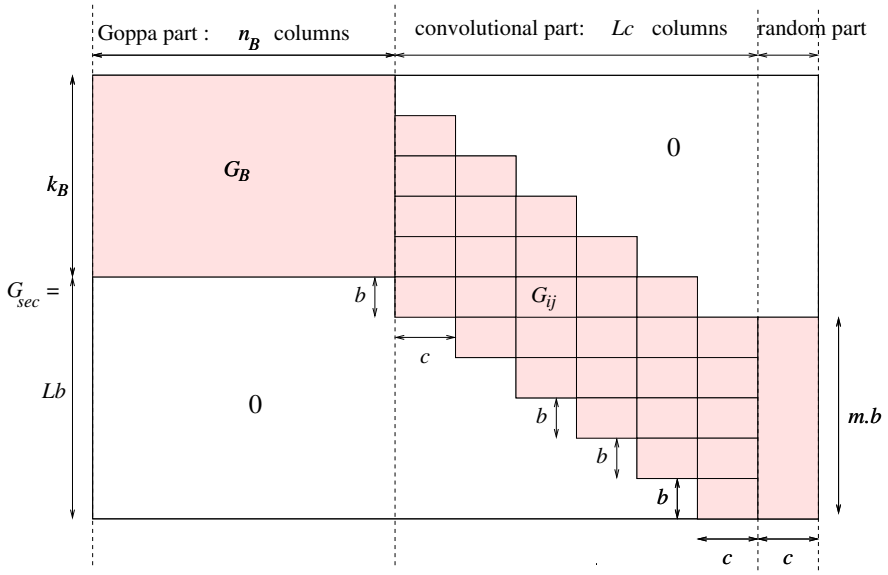


**Fig. 1.** The secret generator matrix. The areas in light pink indicate the only non zero parts of the matrix. $G_B$ is a generator matrix of a binary Goppa code of length $n_B$ and dimension $k_B$. This matrix is concatenated with a matrix of a time varying binary convolutional code where $b$ bits of information are transformed into $c$ bits of data (the corresponding $G_{ij}$ blocks are therefore all of size $b \times c$) and terminated with $c$ random columns at the end. The dimension of the corresponding code is $k \stackrel{\text{def}}{=} k_B + Lb$ and the length is $n \stackrel{\text{def}}{=} n_B + (L+1)c$ where $L$ is the time duration of the convolutional code.

## 3 Description of the Attack

The purpose of this section is to explain the idea underlying our attack which is a message recovery attack taking advantage of a partial key recovery attack. The attack is divided in two main steps. The first step consists in a (partial) key recovery attack aiming at unraveling the convolutional structure. The second part consists in a message recovery attack taking advantage of the fact

that if the convolutional part is recovered, then an attacker can decrypt a message with good probability if he is able to decode a linear code of dimension $k_B$ and length $n_B$ when there are less than $t_B \stackrel{\text{def}}{=} t\frac{n_B}{n}$ errors (this is the average number of errors that the Goppa code has to decode).

### 3.1 Unraveling the convolutional structure

The authors have chosen the parameters of the scheme proposed in [LJ12] so that it remains hard to find low-weight codewords in the dual of the public code $\mathscr{C}_{\text{pub}}$. It is advocated in [LJ12] that in their case the only deviation from a random code is the convolutional structure in terms of low weight parity-checks. For instance, the following parameters are suggested $(n, k) = (1800, 1200)$ and in the construction phase the authors propose to throw away any code who would have parity-checks of weight less than 125. However, the fact that the structure of $\mathscr{C}_{\text{pub}}$ leads in a natural way to low weight codewords is not taken into account. Indeed, we expect many (i.e. about $2^{b-1}$) codewords of weight less than or equal to $c$. This comes from the fact that the subcode of $\mathscr{C}_{\text{pub}}$ generated by the last $b$ rows of $\boldsymbol{G}_{\text{sec}}$ (and permuted by $\boldsymbol{P}$) has support of size $2c$ and dimension $b$. Therefore any algorithm aiming at finding codewords of weight less than $c$ say should output such codewords. Looking at the support of such codewords reveals the $2c$ last columns of $\boldsymbol{G}_{\text{sec}}$. By puncturing these columns and starting this process again but this time by looking for codewords of weight less than $c/2$ (since this time the punctured code contains a subcode of dimension $b$ and support of size $c$ arising from the penultimate block of rows of $\boldsymbol{G}_{\text{sec}}$) will reveal the following block of $c$ columns of the matrix. In other words we expect to capture by these means a first subcode of dimension $b$ and support the $2c$ last positions of $\mathscr{C}_{\text{sec}}$. Then we expect a second subcode of dimension $b$ with support the $3c$ last positions of $\mathscr{C}_{\text{pub}}$ and so on and so forth. Finally we expect to be able after suitable column swapping to obtain the generator matrix $G'$ of an equivalent code to $\mathscr{C}_{\text{pub}}$ which would have the form indicated in Figure 2.

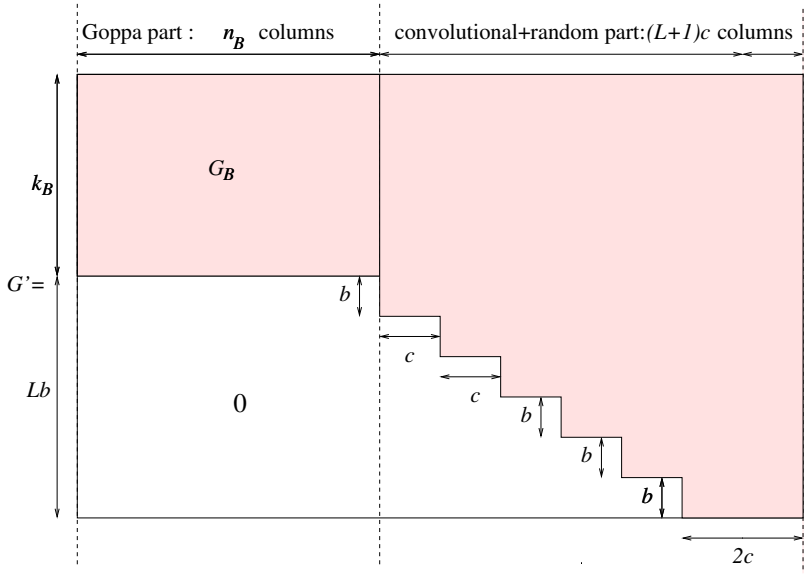


**Fig. 2.** The generator matrix of an equivalent code obtained by our approach.$\boldsymbol{G}'_B$ denotes the generator matrix of a Goppa code which is equivalent to the code with generator matrix $\boldsymbol{G}_B$.

More precisely the algorithm for finding a generator matrix of a code equivalent to $\mathscr{C}_{\text{pub}}$ is given by Algorithm 1 given below.

We assume here that :

**Algorithm 1** An algorithm for finding $\boldsymbol{G}'$.

**input:** $\boldsymbol{G}_{\text{pub}}$ the public generator matrix
**output:** a generator matrix $\boldsymbol{G}'$ of a code equivalent to $\mathscr{C}_{\text{pub}}$ which has the form indicated in Fig. 2.

$\mathcal{L} \leftarrow []$
**for** $i = L, \ldots, 1$ **do**
  $\boldsymbol{G} \leftarrow \texttt{GeneratorMatrixPuncturedCode}(\mathscr{C}_{\text{pub}}, \mathcal{L})$
  $\boldsymbol{G} \leftarrow \texttt{LowWeight}(\boldsymbol{G}, w)$
  $w \leftarrow \texttt{Function}(i)$
  $\boldsymbol{G}_i \leftarrow \texttt{ExtendedGeneratorMatrix}(\boldsymbol{G}, \mathcal{L}, \mathscr{C}_{\text{pub}})$
  $\mathcal{L} \leftarrow \texttt{Support}(\boldsymbol{G}) || \mathcal{L}$
**end for**
$\boldsymbol{G} \leftarrow \texttt{GeneratorMatrixPuncturedCode}(\mathscr{C}_{\text{pub}}, \mathcal{L})$
$\boldsymbol{G}_0 \leftarrow \texttt{ExtendedGeneratorMatrix}(\boldsymbol{G}, \mathcal{L}, \mathscr{C}_{\text{pub}})$
$\boldsymbol{G}'$ is the concatenation of the rows of $\boldsymbol{G}_0, \boldsymbol{G}_1, \ldots, \boldsymbol{G}_L$.
**return** $\boldsymbol{G}'$

- the function `GeneratorMatrixPuncturedCode` takes as input a code $\mathscr{C}$ of length $n$ and an ordered set of positions $\mathcal{L}$ which is a sublist of $[1, 2, \ldots, n]$ and outputs a generator matrix of $\mathscr{C}$ punctured in the positions belonging to $\mathcal{L}$;
- `Function` will be a certain function which will be specified later on;
- `Support`($\mathscr{C}$) yields the (ordered) support of $\mathscr{C}$ and $||$ is the concatenation of lists;
- the function `LowWeight` takes as input a code $\mathscr{C}$ and a weight $w$. It outputs a generator matrix of a subcode of $\mathscr{C}$ obtained by looking for codewords of weight less than or equal to $w$. Basically a certain number of codewords of weight $\leq w$ are produced and the positions which are involved in at least $t$ codewords are put in a list $\mathcal{L}$ (where $t$ is some threshold depending on the weight $w$, the length $n$ of the code, its dimension $k$ and the number of codewords produced by the previous algorithm), which means that $i$ is taken as soon as there are at least $c$ elements in $\mathscr{C}$ for which $c_i = 1$. Then a generator matrix for the subcode of $\mathscr{C}$ formed by the codewords of $\mathscr{C}$ whose coordinates outside $\mathcal{L}$ are all equal to 0 is returned. See Algorithm 2 for further details.
- the function `ExtendedGeneratorMatrix` takes as input a generator matrix of some code $\mathscr{C}'$, an ordered set of positions $\mathcal{L}$ and a code $\mathscr{C}$ such that $\mathscr{C}'$ is the result of the puncturing of $\mathscr{C}$ in the positions belonging to $\mathcal{L}$. It outputs a generator matrix of the permuted subcode $\mathscr{C}''$ of $\mathscr{C}$ whose positions are reordered in such a way that the first positions correspond to the positions of $\mathscr{C}'$ and the remaining positions to the ordered list $\mathcal{L}$. This code $\mathscr{C}''$ corresponds to the codewords of $\mathscr{C}'$ which are extended as codewords of $\mathscr{C}$ over the positions belonging to $\mathcal{L}$ in an arbitrary linear way.

### 3.2 Finishing the job : decoding the code with generator matrix $\boldsymbol{G}'_{\boldsymbol{B}}$

If we are able to decode the code with generator matrix $\boldsymbol{G}'_B$, then standard sequential decoding algorithms for convolutional codes will allow to decode the last $(L+1)c$ positions. Let $\boldsymbol{G}'_B$ be the generator matrix of a code equivalent to the secret Goppa code chosen for the scheme specified in Figure 2. Decoding such a code can be done by algorithms aiming at decoding generic linear codes such as Stern's algorithm [Ste88] and its subsequent improvements [Dum91,BLP11,MMT11,BJMM12]. This can be done for the parameters suggested in [LJ12].

## 4 Implementation of the attack for the parameters suggested in [LJ12]

We have carried out the attack on the parameters suggested in [LJ12]. They are provided in Table **??**.

**Algorithm 2** `LowWeight`$(\boldsymbol{G}, w)$

**input:**

- $\boldsymbol{G}$ a certain $k \times n$ generator matrix of a code $\mathscr{C}$;
- $w$ a certain weight.

**output:** a generator matrix $\boldsymbol{G}'$ of a subcode of $\mathscr{C}$ obtained from the supports of a certain subset of codewords of weight $w$ in $\mathscr{C}$.

$\mathscr{C} \leftarrow$ `LowWeightCodewordSearch`$(\boldsymbol{G}, w)$ {Produces a set of linear combinations of rows of $\boldsymbol{G}$ of weight $\leq w$}
Initialize an array $tab$ of length $n$ to zero
$t \leftarrow$ `Threshold`$(w, n, k, |\mathscr{C}|)$
**for all** $c \in \mathscr{C}$ **do**
  **for** $i \in [1..n]$ **do**
    **if** $c_i = 1$ **then**
      $tab[i] \leftarrow tab[i] + 1$
    **end if**
  **end for**
**end for**
$\mathcal{L} \leftarrow []$
**for** $i \in [1..n]$ **do**
  **if** $tab[i] \geq t$ **then**
    $\mathcal{L} \leftarrow \mathcal{L}||\{i\}$
  **end if**
**end for**
$\boldsymbol{G}' \leftarrow$ `SorthenedCode`$(\boldsymbol{G}, \mathcal{L})$ {Produces a generator matrix for the subcode of $\mathscr{C}$ formed by the codewords of $\mathscr{C}$ whose coordinates outside $\mathcal{L}$ are all equal to 0.}
**return** $\boldsymbol{G}'$.

**Table 1.** Parameters for the second scheme suggested in [LJ12].

| $n$ | $n_B$ | $k$ | $k_B$ | $b$ | $c$ | $L$ | $m$ | $t$ (number of errors) |
|---|---|---|---|---|---|---|---|---|
| 1800 | 1020 | 1160 | 660 | 20 | 30 | 25 | 12 | 45 |

Setting the weight parameter $w$ accurately when calling the function `LowWeight` is the key for finding the 60 last positions. If $w$ is chosen to be too large, for instance when $w = 22$, running Dumer's low weight codeword search algorithm [Dum91] gave the result given in Figure 3 concerning the frequencies of the code positions involved in the codewords of weight less than 22 output by the algorithm and stored in table $tab$ during the execution of the algorithm.

We see in Figure 3 that this discriminates the 90 last code positions and not as we want the 60 last code positions. However choosing $w$ to be equal to 18 enables to discriminate the 60 last positions as shown in Figure 4.

Data used in Figure 4 come from 3900 codewords generated in one hour and a half on an Intel Xeon W3550 (3 GHz) CPU by a monothread implementation in C of Dumer's algorithm. The message recovery part of the attack involving the Goppa code consists in decoding 25.5 errors on average in a linear code of dimension 660 and length 1020. The time complexity is about $2^{42}$. This second part of the attack could be achieved using the previous program on the same computer in about 6.5 hours on average.

## 5 Analysis of the security of the scheme

### 5.1 An improved attack

The purpose of this section is to provide a very crude analysis of the security of the scheme. We will not analyze our attack detailed in Section 3, since even if it was enough to break the second

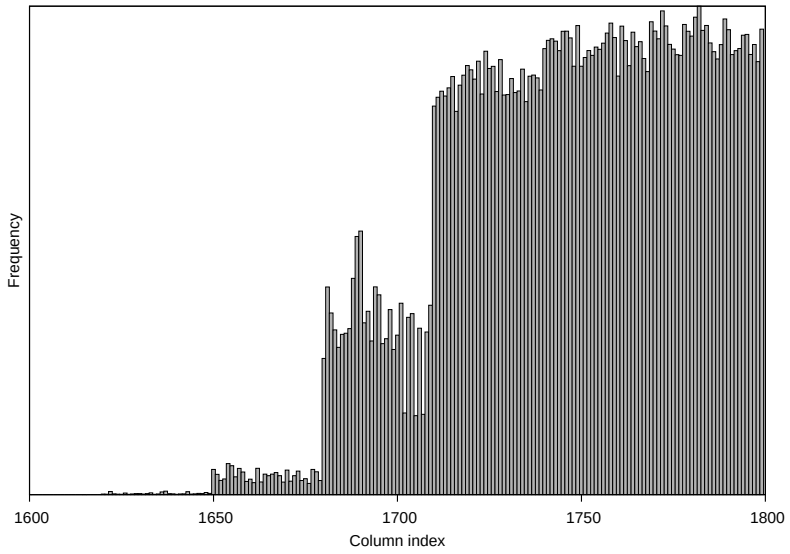


**Fig. 3.** The frequencies of the code positions involved in codewords of weight $\leq 22$ output by Dumer's algorithm.

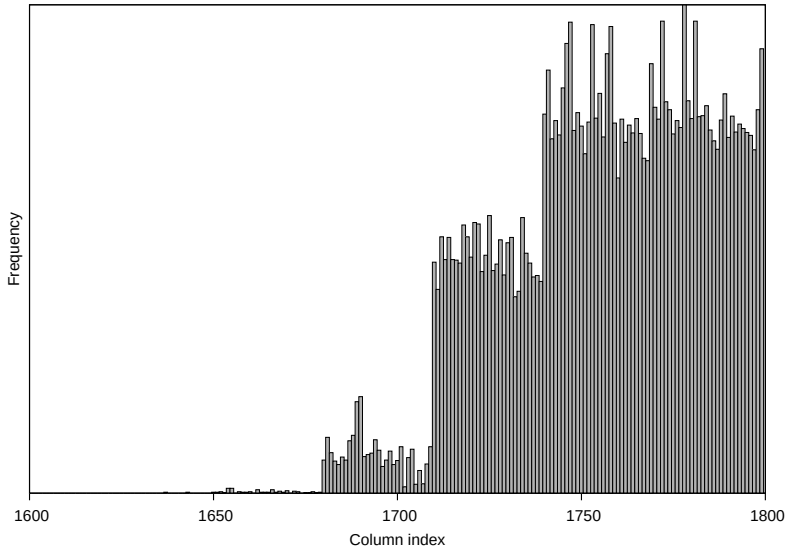


**Fig. 4.** The frequencies of the code positions involved in codewords of weight $\leq 18$ output by Dumer's algorithm.

scheme suggested in [LJ12] it is not the most efficient one. We will give a sketch of a better attack and a rough analysis for it. Basically, the real threat on this scheme comes from the fact that there

exists a subcode $\mathscr{C}$ of $\mathscr{C}_{\text{pub}}$ of very small support (of size $2c$ here), namely the code generated by the last $b$ rows of $\boldsymbol{G}_{\text{sec}}$ permuted by the secret permutation matrix $\boldsymbol{P}$. For instance, there are about $2^{b-1}$ codewords of weight less than or equal to $c$ which should be found by a low weight codeword searching algorithm and which should reveal the support of $\mathscr{C}$. This is basically the idea underlying our attack. However there are other subcodes of rather small support which yield low weight codewords, namely the codes $\mathscr{C}_s$ generated by the $s \times b$ last rows of $\boldsymbol{G}$ for $s$ ranging between 2 and $L$. The support of $\mathscr{C}_s$ is of size $(s+1)c$. Notice that its rate gets closer and closer to the rate $\frac{2}{3}$ (which is more or less the rate of the final code) as $s$ increases. This is a phenomenon which helps low weight codeword algorithms as will be explained later on.

An improvement of our attack would consist in using a low weight codeword algorithm in order to find one of the codewords of $\mathscr{C}_s$ and to use this codeword $c$ to bootstrap from here to find the whole support of $\mathscr{C}_s$. This is very much in the spirit of the attack against the KKS scheme which is explained in Algorithm 2 which can be found in Subsection 4.4 of [OT11]. With this approach, by using the codeword which has already been found, it is much easier to find new ones belonging to the same subcode with small support by imposing that the information set used for finding low weight codewords is chosen outside the positions belonging to the support of $c$. The complexity of the whole attack is dominated in this case by the complexity of finding just one codeword in $\mathscr{C}$ when there is a good way to identify the candidates in $\mathscr{C}$ (which can be done by checking the weight of $c$). Notice that it is very likely that $\mathscr{C}$ is actually the sub code of $\mathscr{C}$ of dimension $b$ which has the smallest support. Recall here that this is precisely the notion captured by the generalized Hamming weights of a code [WY93], $w_i$ being defined as the smallest support of a subcode of dimension $i$. In other words $w_1$ is nothing but the minimum distance of the code and in our case it is likely that $w_b = 2c$ (and more generally $w_{sb} = (s+1)c$ for $s = 1..L$). In other words, the problem which should be difficult to solve is the following one

*Problem 1.* Find one of the subcodes of dimension $s \times b$ whose support size is the $s \times b$-th generalized Hamming weight of $\mathscr{C}_{\text{pub}}$.

We will focus now on the following approach to solve this problem. Consider a low weight codeword algorithm which aims at finding low weight codewords in a code of dimension $k$ by picking up a random set of positions $\mathcal{I}$ of size slightly larger than $k$, say $k+l$ and which looks for all (or at least a non-negligible fraction) of codewords which have weight equal to some small quantity $p$ over these positions. These quantities are very good candidates for having low weight over the whole support. This is precisely the approach which is followed in the best low weight codeword search algorithms such as [Ste88,Dum91,MMT11,BJMM12]. We run such an algorithm for several different sets $\mathcal{I}$ and will be interested in the complexity of outputting at least one codeword which belongs to $\mathscr{C}$. This is basically the approach which has been very successful to break the KKS scheme [OT11] and which is the natural candidate to break the [LJ12] scheme.

To analyze such an algorithm we will make some simplifying assumptions

- The cost of checking one of those $\mathcal{I}$ is of order $O\left(L + \frac{L^2}{2^l}\right)$ where $L = \sqrt{\binom{k+l}{p}}$. We neglect here the cost coming from writing the parity-check matrix in systematic form and this does not really cover the recent improvements in [MMT11,BJMM12]. We have made here such an approximation for sake of simplicity. We refer to [FS09] for an explanation of this cost.
- We assume that the result of the puncturing of $\mathscr{C}$ by all positions which do no belong to its support behaves like a random code of dimension $k'$ and length $n'$.

Our main result to analyze such an algorithm consists in the following proposition.

**Proposition 1.** *Let*

- $f(x)$ *be the function defined by* $f(x) \stackrel{def}{=} \max\left(x(1-x/2), 1-\frac{1}{x}\right)$ ;
- $p(s) \stackrel{def}{=} \frac{\binom{n'}{s}\binom{n-n'}{k+l-s}}{\binom{n}{k+l}}$;
- $\lambda(s) \stackrel{def}{=} \binom{s}{p} 2^{k'-s}$.

– $C(k,l,p) \stackrel{def}{=} L + \frac{L^2}{2^l}$ *where* $L \stackrel{def}{=} \sqrt{\binom{k+l}{p}}$;
– $P \stackrel{def}{=} \sum_{s=1}^{n'} p(s) f(\lambda)$.

*Then the complexity that the low weight codeword search algorithm outputs an element in* $\mathscr{C}$ *is of order*

$$O\left(\frac{C(k,l,p)}{P}\right).$$

### 5.2 Proof of Proposition 1

Our first ingredient is a lower bound on the probability that a given set $X$ intersects a random linear code $\mathscr{C}_{\rm rand}$ of dimension $k$ and length $n$ picked up uniformly at random. This lemma gives a sharp lower bound even when $X$ is very large and when there is a big gap between the quantities $\mathbf{prob}(X \cap \mathscr{C}_{\rm rand} \neq \emptyset) = \mathbf{prob}(\cup_{x \in X} \{x \in \mathscr{C}_{\rm rand}\})$ and $\sum_{x \in X} \mathbf{prob}(x \in \mathscr{C}_{\rm rand})$.

**Lemma 1.** *Let* $X$ *be some subset of* $\mathbb{F}_2^n$ *of size* $m$ *and let* $f$ *be the function defined by* $f(x) \stackrel{def}{=} \max\left(x(1-x/2), 1-\frac{1}{x}\right)$. *We denote by* $x$ *the quantity* $\frac{m}{2^{n-k}}$, *then*

$$\mathbf{prob}(X \cap \mathscr{C}_{rand} \neq \emptyset) \geq f(x).$$

This lemma can be found in [OT11] and it is proved there.
Let us finish now the proof of Proposition 1. Denote by $\mathcal{J}$ the support of $\mathscr{C}$:

$$\mathcal{J} \stackrel{\rm def}{=} \mathsf{supp}(\mathscr{C}).$$

Let us first calculate the expected number of sets $\mathcal{I}$ we have to consider before considering an element of $\mathscr{C}$. Such an event happens precisely when there is a nonzero word in $\mathscr{C}$ whose restriction to $\mathcal{I} \cap \mathcal{J}$ is of weight equal to $p$. Let $\mathscr{C}_{\mathcal{I} \cap \mathcal{J}}$ be the restriction of the codewords of $\mathscr{C}$ to the positions which belong to $\mathcal{I} \cap \mathcal{J}$, that is

$$\mathscr{C}_{\mathcal{I} \cap \mathcal{J}} \stackrel{\rm def}{=} \{(c_i)_{i \in \mathcal{I} \cap \mathcal{J}} : (c_i)_{1 \leq i \leq n} \in \mathscr{C}\}.$$

Let $X$ be the set of non-zero binary words of support $\mathcal{I} \cap \mathcal{J}$ which have weight equal to $p$. Denote by $W$ the size of $\mathcal{I} \cap \mathcal{J}$. The probability that $W$ is equal to $s$ is precisely

$$\mathbf{prob}(W = s) = \frac{\binom{n'}{s}\binom{n-n'}{k+l-s}}{\binom{n}{k+l}} = p(s).$$

Then the probability $P$ that a certain choice of $\mathcal{I}$ gives among the codewords considered by the algorithm a codeword of $\mathscr{C}$ can be expressed as

$$P = \sum_{s=1}^{n'} \mathbf{prob}(W = s) \mathbf{prob}(X \cap \mathscr{C}' \neq \emptyset) \tag{1}$$

$$\geq \sum_{s=1}^{n'} p(s) f(\lambda) \tag{2}$$

by using Lemma 1 with $\mathscr{C}'$ and the aforementioned $X$. Therefore the average number of iterations which have to be performed before finding an element in $\mathscr{C}$ is equal to $\frac{1}{P}$ and this yields immediately Proposition 1.

### 5.3 Repairing the parameters and a pitfall

A possible way to repair the scheme consists in increasing the size of the random part (which corresponds to the last $c$ columns in $\boldsymbol{G}_{\text{sec}}$ here). Instead of choosing this part to be of size $c$ as suggested in [LJ12], its size can be increased in order to thwart the algorithm of Subsection 5.1. Let $r$ be the number of random columns we add at the end of the convolutional part, so that the final length of the code is now $n_B + Lc + r$ instead of $n_B + (L+1)c$ as before. If we choose $r$ to be equal to 140, then the aforementioned attack needs about $2^{80}$ operations before outputting an element of $\mathscr{C}$ which is the (permuted) subcode corresponding to the last $b$ rows of $\boldsymbol{G}_{\text{sec}}$. As before, let us denote by $\mathscr{C}_s$ the permuted (by $\boldsymbol{P}$) subcode of $\mathscr{C}_{\text{pub}}$ generated by the last $s \times b$ rows of $\boldsymbol{G}_{\text{sec}}$ permuted by $\boldsymbol{P}$. We can use the previous analysis to estimate the complexity of obtaining an element of $\mathscr{C}_s$ by the previous algorithm. We have gathered the results in Table 2.

**Table 2.** Complexity of obtaining at least one element of $\mathscr{C}_s$ by the algorithm of Subsection 5.1

| $s$ | 1 | 5 | 10 | 15 | 20 | 21 | 22 | 25 |
|---|---|---|---|---|---|---|---|---|
| complexity (bits) | 80.4 | 72.1 | 65.1 | 61.0 | 59.4 | 59.3 | 59.4 | 59.8 |

We see from this table that in this case the most important threat does not come from finding low weight codewords arising from codewords in $\mathscr{C}_1$, but codewords of moderate weight arising from codewords in $\mathscr{C}_{20}$ for instance. Codewords in this code have average weight $\frac{r+20c}{2} = 370$. This implies that a simple policy for detecting such candidates which consists in keeping all the candidates in the algorithm of Subsection 5.1 which have weight less than this quantity is very likely to filter out the vast majority of bad candidates and keep with a good chance the elements of $\mathscr{C}_{20}$. Such candidates can then be used as explained in Subsection 5.1 to check whether or not they belong to a subcode of large dimension and small support.

There is a simple way for explaining what is going on here. Notice that the rate of $\mathscr{C}$ is equal to $\frac{b}{c+r}$, which is much smaller than the rate of the overall scheme which is close to $\frac{b}{c}$ in this case by the choice of the parameters of the Goppa code. However as $s$ increases, the rate of $\mathscr{C}_s$ gets closer and closer to $\frac{b}{c}$, since its rate is given by $\frac{sb}{sc+r} = \frac{b}{c+r/s}$. Assume for one moment that the rate of $\mathscr{C}_s$ is equal to $\frac{b}{c}$. Then putting $\boldsymbol{G}_{\text{pub}}$ in systematic form (which basically means that we run the aforementioned algorithm with $p = 1$ and $l = 0$) is already likely to reveal most of the support of $\mathscr{C}_s$ by looking at the support of the rows which have weight around $\frac{sc+r}{2}$ (notice that this phenomenon was already observed in [Ove07]). This can be explained like this. We choose $\mathcal{I}$ to be of size $k$, the dimension of $\mathscr{C}_{\text{pub}}$, and to be an information set for $\mathscr{C}_{\text{pub}}$. Then, because the rate of $\mathscr{C}_s$ is equal to the rate of $\mathscr{C}_{\text{pub}}$, we expect that the size of $\mathcal{I} \cap \mathcal{J}$ (where $\mathcal{J}$ is the support of $\mathscr{C}_s$) has a rather good chance to be of size smaller than or equal to the dimension of $\mathscr{C}_s$. This in turn implies that it is possible to get codewords from $\mathscr{C}_s$ by any choice over the information set $\mathcal{I}$ of weight 1 which is non zero over $\mathcal{I} \cap \mathcal{J}$ (and therefore of weight 1 there). More generally, even if $\mathcal{I} \cap \mathcal{J}$ is slightly bigger than the dimension of $\mathscr{C}_s$ we expect to be able to get codewords in $\mathscr{C}_s$ a soon as $p$ is greater than the Gilbert-Varshamov distance of the restriction $\mathscr{C}'_s$ of $\mathscr{C}_s$ to $\mathcal{I} \cap \mathcal{J}$, because there is in this case a good chance that this punctured code has codewords of weight $p$. This Gilbert-Varshamov distance will be very small in this case, because the rate of $\mathscr{C}_s$ is very close to 1 (it is expected to be equal to $\frac{\dim(\mathscr{C}_s)}{|\mathcal{I} \cap \mathcal{J}|}$).

Nevertheless, it is clear that it should be possible to set up the parameters (in particular increasing $r$ should do the job) so that existing low weight codeword algorithms should be unable to find these subcodes $\mathscr{C}_s$ with complexity less than some fixed threshold. However, all these codes $\mathscr{C}_s$ have to be taken into account and the attacks on the dual have also to be reconsidered carefully ([LJ12] considered only attacks on the dual aiming at finding the codewords of lowest weight, but obviously the same technique used for finding some of the $\mathscr{C}_s$ will also work for the dual). Moreover, even if by construction the restriction of $\mathscr{C} = \mathscr{C}_1$ to its support should behave as a random code, this is not true anymore for $\mathscr{C}_s$ with $s$ greater than one, due to the convolutional structure. The

analysis sketched in Subsection 5.1 should be adapted a little bit for this case and should take into account the improvements over low weight searching algorithms [MMT11,BJMM12]. Finally, setting up the parameters also requires a careful study of the error probability that sequential decoding fails. This whole thread of work is beyond the scope of the present paper.

## References


[BBC08] M. Baldi, M. Bodrato, and G.F. Chiaraluce. A new analysis of the McEliece cryptosystem based on QC-LDPC codes. In *Security and Cryptography for Networks (SCN)*, pages 246–262, 2008.

[BBC+11] M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, and D. Schipani. Enhanced public key security for the McEliece cryptosystem. submitted, 2011. arxiv:1108.2462v2[cs.IT].

[BBD09] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors. *Post-Quantum Cryptography*. Springer-Verlag, 2009.

[BC07] Marco Baldi and Franco Chiaraluce. Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pages 2591 –2595, june 2007.

[BCGO09] 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.

[BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In *Eurocrypt 2012*, LNCS. Springer, 2012.

[BL05] T. P. Berger and P. Loidreau. How to mask the structure of codes for a cryptographic use. *Designs Codes and Cryptography*, 35(1):63–79, 2005.

[BLP11] D. J. Bernstein, T. Lange, and C. Peters. Smaller decoding exponents: ball-collision decoding. In *Proceedings of Crypto 2011*, volume 6841 of *LNCS*, pages 743–760, 2011.

[CFS01] N. Courtois, M. Finiasz, and N. Sendrier. How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology – Asiacrypt'2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174, Gold Coast, Australia, 2001. Springer.

[CGG+12] A. Couvreur, P. Gaborit, V. Gauthier, A. Otmani, and J.P. Tillich. Distinguisher-based attacks on public-key cryptosystems using reed-solomon codes. submitted to WCC 2013, December 2012.

[Dum91] I. Dumer. On minimum distance decoding of linear codes. In *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, pages 50–52, Moscow, 1991.

[FGO+] J.-C. Faugère, V. Gauthier, A. Otmani, L. Perret, and J.-P. Tillich. A distinguisher for high rate McEliece cryptosystems.

[FGO+10] Jean-Charles Faugère, Valérie Gauthier, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. A distinguisher for high rate McEliece cryptosystems. Cryptology ePrint Archive, Report 2010/331, 2010. http://eprint.iacr.org/.

[FGO+11] J.-C. Faugère, V. Gauthier, A. Otmani, L. Perret, and J.-P. Tillich. A distinguisher for high rate McEliece cryptosystems. In *Proceedings of the Information Theory Workshop 2011, ITW 2011*, pages 282–286, Paraty, Brasil, 2011.

[FM08] C. Faure and L. Minder. Cryptanalysis of the McEliece cryptosystem over hyperelliptic curves. In *Proceedings of the eleventh International Workshop on Algebraic and Combinatorial Coding Theory*, pages 99–107, Pamporovo, Bulgaria, June 2008.

[FS09] M. Finiasz and N. Sendrier. Security bounds for the design of code-based cryptosystems. In M. Matsui, editor, *Asiacrypt 2009*, volume 5912 of *LNCS*, pages 88–105. Springer, 2009.

[GOT12] V. Gauthier, A. Otmani, and J.-P. Tillich. A distinguisher-based attack on a variant of McEliece's cryptosystem based on Reed-Solomon codes. *CoRR*, abs/1204.6459, 2012.

[JM96] H. Janwa and O. Moreno. McEliece public key cryptosystems using algebraic-geometric codes. *Designs Codes and Cryptography*, 8(3):293–307, 1996.

[KKS97] G. Kabatianskii, E. Krouk, and B. J. M. Smeets. A digital signature scheme based on random error-correcting codes. In *IMA Int. Conf.*, volume 1355 of *Lecture Notes in Computer Science*, pages 161–167. Springer, 1997.

[Kra91] D. Kravitz. Digital signature algorithm. US patent 5231668, July 1991.

[LJ12] C. Löndahl and T. Johansson. A new version of McEliece PKC based on convolutional codes. In *ICICS*, pages 461–470, 2012.

[MB09] 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.

[McE78] R. J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $O(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Asiacrypt 2011*, volume 7073 of *LNCS*, pages 107–124. Springer, 2011.

[MS07] L. Minder and A. Shokrollahi. Cryptanalysis of the Sidelnikov cryptosystem. In *Eurocrypt 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 347–360, Barcelona, Spain, 2007.

[MTSB12] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. L. M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. *IACR Cryptology ePrint Archive*, 2012:409, 2012.

[Nie86] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.

[OT11] A. Otmani and J.-P. Tillich. An efficient attack on all concrete KKS proposals. In *PQCrypto*, pages 98–116, 2011.

[OTD10] A. Otmani, J.P. Tillich, and L. Dallot. Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. *Special Issues of Mathematics in Computer Science*, 3(2):129–140, January 2010.

[Ove07] R. Overbeck. Recognizing the structure of permuted reducible codes. In J.P. Tillich D. Augot, N. Sendrier, editor, *proceedings of WCC'2007*, pages 269–276, 2007.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[Sid94] V.M. Sidelnikov. A public-key cryptosytem based on Reed-Muller codes. *Discrete Mathematics and Applications*, 4(3):191–207, 1994.

[SS92] V.M. Sidelnikov and S.O. Shestakov. On the insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discrete Mathematics and Applications*, 1(4):439–444, 1992.

[Ste88] J. Stern. A method for finding codewords of small weight. In G. D. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1988.

[Wie10] Christian Wieschebrink. Cryptanalysis of the niederreiter public key scheme based on grs subcodes. In *PQCrypto*, pages 61–72, 2010.

[WY93] V. K.-W. Wei and K. Yang. On the generalized Hamming weights of product codes. *Trans. Inf. Theory*, 39(5):1709–1713, 1993.