A New Analysis of the McEliece Cryptosystem Based on QC-LDPC Codes[⋆]

Marco Baldi[1], Marco Bodrato[2], and Franco Chiaraluce[1]

[1] DEIT, Università Politecnica delle Marche
Ancona, Italy
{m.baldi,f.chiaraluce}@univpm.it

[2] Centro "Vito Volterra", Università di Roma Tor Vergata
Roma, Italy
bodrato@mail.dm.unipi.it

Abstract. We improve our proposal of a new variant of the McEliece cryptosystem based on QC-LDPC codes. The original McEliece cryptosystem, based on Goppa codes, is still unbroken up to now, but has two major drawbacks: long key and low transmission rate. Our variant is based on QC-LDPC codes and is able to overcome such drawbacks, while avoiding the known attacks. Recently, however, a new attack has been discovered that can recover the private key with limited complexity. We show that such attack can be avoided by changing the form of some constituent matrices, without altering the remaining system parameters. We also propose another variant that exhibits an overall increased security level. We analyze the complexity of the encryption and decryption stages by adopting efficient algorithms for processing large circulant matrices. The Toom-Cook algorithm and the short Winograd convolution are considered, that give a significant speed-up in the cryptosystem operations.

Key words: McEliece cryptosystem, QC-LDPC codes, Cryptanalysis, Toom-Cook, Winograd.

1 Introduction

The McEliece cryptosystem is a public-key cryptosystem based on algebraic coding theory [1] that revealed to have a very high security level. It adopts a generator matrix as the private key and one transformation of it as the public key, while its security lies in the difficulty of decoding a large linear code with no visible structure, that is known to be an NP complete problem [2].

The original McEliece cryptosystem is still unbroken, in the sense that a total break attack has never been found, and even local deduction attacks remain quite intractable in practice [3,4]. Moreover, the system is two or three orders of magnitude faster than competing solutions, like RSA, that is among the most popular public key algorithms currently in use. Despite this, the McEliece cryptosystem has been rarely considered in practical applications; this is due to the fact it exhibits two major drawbacks: i) large size of the public key and ii) low transmission rate (that is about 0.5).

[⋆] © Springer 2008. LNCS 5229, "6th International Conference SCN 2008, Amalfi, Italy"

In his original formulation, McEliece used Goppa codes of length $n = 1024$, dimension $k = 524$, and minimum distance $d_{\min}$ of at least 101, that are able to correct $t = 50$ errors. Several attempts have been made, later on, for overcoming the drawbacks of the original system and/or further reducing the complexity, but the adoption of alternative families of codes has not been possible without compromising the system security. Generalized Reed-Solomon (GRS) codes, in particular, were initially considered for an important variant of the McEliece cryptosystem, proposed by Niederreiter [5], but they revealed to be unsecure. On the other hand, when employing Goppa codes, the Niederreiter cryptosystem shows some advantages: it has equivalent security to that of the McEliece system, for codes with the same parameters [6], but it requires a key size more than halved (when considering the values reported above), a transmission rate slightly increased, and the possibility to use a public key in systematic form. For these reasons, though renouncing to use GRS codes, the Niederreiter system is considered a good alternative to the McEliece system.

A clever technique for increasing the transmission rate has been proposed by Riek [7], and consists in mapping additional information bits onto the intentional error vector. This approach could increase significantly the transmission rate, but requires the introduction of an additional encoding and decoding stage to implement a positional code on error vectors.

Low-Density Parity-Check (LDPC) codes represent the state of the art in forward error correction techniques, and permit to approach the theoretical Shannon limit [8], while ensuring limited complexity. Quasi-cyclic (QC) LDPC codes are a particular class of LDPC codes, able to join low complexity encoding of QC codes with high-performing and low-complexity decoding techniques based on the belief propagation principle. Several classes of QC-LDPC codes have been proposed up to now, all having in common the parity-check matrix structure, that is formed by sparse circulant blocks.

In a recent work, we have proposed to adopt a particular family of QC-LDPC codes in the McEliece cryptosystem to reduce the key size and increase the transmission rate [9]. We have shown such variant is able to counter all the general attacks, and even new attacks that can compromise the security of previous LDPC-based versions of the cryptosystem, like that proposed in [10].

Very recently, however, Otmani, Tillich and Dallot developed a new attack that, exploiting a flaw in the transformation from the private key to the public key, is able to recover the secret key with very high probability [11]. They presented three attack strategies, that will be denoted as OTD1, OTD2 and OTD3 in the following. In the same work, the authors also proved that a previous proposal for the adoption of quasi-cyclic (but not LDPC) codes for shortening the public key of the McEliece cryptosystem [12] is not secure.

In this paper, we shortly describe the three OTD attacks, and analyze the flaw in the private-public key map that originates them. We propose a first variant of the cryptosystem that is able to counter such attacks by adopting a different form for its constituent matrices, without altering other parameters.

Furthermore, we present a second variant of the cryptosystem that provides overall increased security.

In addition, we study the application, in this new version of the cryptosystem, of efficient algorithms for computation on large circulant matrices, that permit to reduce its encryption and decryption complexity. The main question concerns encoding, that, in the case of QC codes, can be implemented through a barrel shift-register with hardware complexity that increases linearly with the code length. However, the total number of operations can still be high, thus reflecting in latency issues. A common solution to this problem consists in searching for sparse generator matrices [13,14]. Though this is possible for suitably designed codes, such approach is unsuitable for codes randomly designed for the use in cryptographic systems. In this case, however, efficient computation algorithms can limit the number of operations. In this paper, we consider two possible choices of efficient multiplication algorithms, namely, the Toom-Cook algorithm and the short Winograd convolution, that actually yield reduced complexity.

2 Notation

Let $\mathbb{F} = GF(m)$ be the Galois field of order m, with m a prime power.

A $p \times p$ Toeplitz matrix $\mathbf{A}$ over $\mathbb{F}$ is defined as follows:

$$\mathbf{A} = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{p-1} \\ a_{-1} & a_0 & a_1 & \cdots & a_{p-2} \\ a_{-2} & a_{-1} & a_0 & \cdots & a_{p-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{1-p} & a_{2-p} & a_{3-p} & \cdots & a_0 \end{bmatrix} . \tag{1}$$

It is called circulant if $\forall i, a_i = a_{i-p}$. In this work we restrict our analysis to binary circulant matrices, that is, we consider $m = 2$.

There is a natural isomorphism between the algebra of $p \times p$ circulant matrices with entries in the field $\mathbb{F}$ and the ring of polynomials $\mathbb{F}[x]/(x^p + 1)$. If we denote by $\mathbf{X}$ the matrix with entries:

$$X_{i,j} = \begin{cases} 1 \text{ if } j - i \equiv 1 \pmod{p} \\ 0 \text{ if } j - i \not\equiv 1 \pmod{p} \end{cases} , \tag{2}$$

the isomorphism maps the matrix $\mathbf{X}$ to the monomial x and the generic circulant matrix $\sum_{i=0}^{p-1} \alpha_i \mathbf{X}^i$ to the polynomial $\sum_{i=0}^{p-1} \alpha_i x^i \in \mathbb{F}[x]/(x^p + 1)$. This isomorphism can be extended to matrices with circulant blocks, as will be shown in the next section.

3 Improved McEliece Cryptosystem Based on QC-LDPC Codes

In order to hide the secret code's structure, we have recently proposed a variant of the McEliece cryptosystem working as follows. Bob, in order to receive encrypted messages, randomly chooses a code in a family of (n_0, d_v, p) QC-LDPC codes based on Random Difference Families [9], by selecting its parity-check matrix $\mathbf{H}$, and produces a generator matrix $\mathbf{G}$ in reduced echelon form. Matrix $\mathbf{H}$ is formed by a row $\{\mathbf{H}_0, \ldots, \mathbf{H}_{n_0-1}\}$ of n_0 binary circulant blocks with size p and row/column weight d_v, and it is part of Bob's private key. Matrix $\mathbf{G}$, instead, is formed by a $k \times k$ identity matrix $\mathbf{I}$ (with $k = k_0 \cdot p$ and $k_0 = n_0 - 1$), followed by a column of k_0 binary circulant blocks with size p. If we suppose $\mathbf{H}_{n_0-1}$ to be non-singular, $\mathbf{G}$ can be obtained as follows:

$$\mathbf{G} = \left[\mathbf{I} \begin{array}{c} \left(\mathbf{H}_{n_0-1}^{-1} \cdot \mathbf{H}_0\right)^T \\ \left(\mathbf{H}_{n_0-1}^{-1} \cdot \mathbf{H}_1\right)^T \\ \vdots \\ \left(\mathbf{H}_{n_0-1}^{-1} \cdot \mathbf{H}_{n_0-2}\right)^T \end{array} \right] . \tag{3}$$

The remaining part of Bob's private key is formed by two other matrices: a $k \times k$ non-singular matrix $\mathbf{S}$ and a sparse $n \times n$ non-singular matrix $\mathbf{Q}$. $\mathbf{S}$ and $\mathbf{Q}$ are regular matrices, formed by $k_0 \times k_0$ and $n_0 \times n_0$ blocks of $p \times p$ binary circulants, respectively. $\mathbf{Q}$ has row/column weight m.

By exploiting the isomorphism introduced in Section 2, the generator matrix $\mathbf{G}$ can be seen as a $k_0 \times n_0$ matrix with entries in the ring of polynomials $\mathbb{R} = GF(2)[x]/(x^p + 1)$; $\mathbf{S}$ and $\mathbf{Q}$ are invertible matrices on the same ring with size $k_0 \times k_0$ and $n_0 \times n_0$, respectively.

Then, Bob computes the public key as follows:

$$\mathbf{G}' = \mathbf{S}^{-1} \cdot \mathbf{G} \cdot \mathbf{Q}^{-1} . \tag{4}$$

It should be noted that $\mathbf{G}'$ preserves the quasi-cyclic structure of $\mathbf{G}$, due to the block circulant form of $\mathbf{S}$ and $\mathbf{Q}$.

$\mathbf{G}'$ is made available in a public directory. Alice, who wants to send an encrypted message to Bob, extracts $\mathbf{G}'$ from the public directory and divides her message into k-bit blocks. If $\mathbf{u}$ is one of these blocks, Alice obtains the encrypted version as follows:

$$\mathbf{x} = \mathbf{u} \cdot \mathbf{G}' + \mathbf{e} = \mathbf{c} + \mathbf{e} .$$

In this expression, $\mathbf{e}$ is a vector of intentional errors randomly generated, with length n and weight t'.

When Bob receives the encrypted message $\mathbf{x}$, he first computes:

$$\mathbf{x}' = \mathbf{x} \cdot \mathbf{Q} = \mathbf{u} \cdot \mathbf{S}^{-1} \cdot \mathbf{G} + \mathbf{e} \cdot \mathbf{Q} . \tag{5}$$

Vector $\mathbf{x}'$ is a codeword of the LDPC code chosen by Bob (corresponding to the information vector $\mathbf{u}' = \mathbf{u} \cdot \mathbf{S}^{-1}$), affected by the error vector $\mathbf{e} \cdot \mathbf{Q}$, whose maximum weight is $t = t' \cdot m$. If t' and m are suitably chosen, Bob is able to correct all the errors with very high probability, by means of LDPC decoding, thus recovering $\mathbf{u}'$, and then $\mathbf{u}$ through a post-multiplication by $\mathbf{S}$.

In the version of QC-LDPC based McEliece cryptosystem proposed in [9], we fixed $n_0 = 4$, $d_v = 13$, $p = 4032$, $m = 7$ and $t' = 27$. Both $\mathbf{S}$ and $\mathbf{Q}$ were chosen sparse, and their non-null circulant blocks had row/column weight equal to m. In particular, the following block-diagonal form for $\mathbf{Q}$ was adopted:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Q}_{n_0-1} \end{bmatrix} \tag{6}$$

that, jointly with its low density, gives raise to the flaw exploited by the OTD attack, shortly described in the following subsection. In addition, as pointed out in [11], matrix $\mathbf{S}$ should contain some null blocks in order to be non-singular.

3.1 The OTD Attack

The adoption of a sparse $\mathbf{S}$ and a sparse block-diagonal $\mathbf{Q}$ implies that, by simply selecting the first k columns of the public key $\mathbf{G}'$, obtained through the transformation (4) with $\mathbf{G}$ in the form (3), an eavesdropper can derive the following matrix:

$$\mathbf{G}'_{\leq k} = \mathbf{S}^{-1} \cdot \begin{bmatrix} \mathbf{Q}_0^{-1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_1^{-1} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{Q}_{n_0-2}^{-1} \end{bmatrix} . \tag{7}$$

By calculating the inverse of $\mathbf{G}'_{\leq k}$ and considering its circulant block at position (i, j), the eavesdropper can easily obtain $\mathbf{Q}_i\mathbf{S}_{i,j}$, being $\mathbf{S}_{i,j}$ the circulant block at position (i, j) in matrix $\mathbf{S}$. Because of the isomorphism, this matrix corresponds to the polynomial:

$$g_{i,j}(x) = q_i(x) \cdot s_{i,j}(x) \bmod (x^p + 1) \tag{8}$$

where polynomials $q_i(x)$ and $s_{i,j}(x)$, in turn, correspond to the blocks $\mathbf{Q}_i$ and $\mathbf{S}_{i,j}$, respectively.

Due to the fact that both $\mathbf{Q}_i$ and $\mathbf{S}_{i,j}$ are sparse (they have row/column weight m), the vector of coefficients of $g_{i,j}(x)$ is obtained as the cyclic convolution of two sparse vectors containing the coefficients of $q_i(x)$ and $s_{i,j}(x)$. For this reason, it is highly probable that $g_{i,j}(x)$ has exactly m^2 non-null coefficients, and its support contains at least one shift $x^{l_a} \cdot q_i(x)$, $0 \leq l_a \leq p - 1$ [11]. This is the initial point for three different attack strategies that, starting from the knowledge of $g_{i,j}(x)$, aim at revealing part of the secret key. They are briefly described next.

OTD1 Attack Strategy. The first attack strategy consists in enumerating all the m-tuples that belong to the support of $g_{i,j}(x)$. Each m-tuple is then validated through inversion of its corresponding polynomial and multiplication by $g_{i,j}(x)$. If the resulting polynomial has exactly m non-null coefficients, the considered m-tuple corresponds to a shifted version of $q_i(x)$ with very high probability. For the specified numerical values, this attack requires a work factor of $2^{50.3}$ binary operations.

OTD2 Attack Strategy. The second attack strategy is based on the periodic autocorrelation of the coefficients vector of $g_{i,j}(x)$. In fact, it is highly probable that the Hadamard product of the polynomial $g_{i,j}(x)$ with a d-shifted version of itself, $g_{i,j}^{d}(x) * g_{i,j}(x)$, results in a shifted version of $q_i(x)$, for a specific value of d. For this reason, the eavesdropper can calculate all the possible $g_{i,j}^{d}(x) * g_{i,j}(x)$ and check whether the resulting polynomial has support with weight m. This attack requires a work factor of 2^{36} binary operations, that could be even further reduced by calculating the periodic autocorrelation of the coefficients of $g_{i,j}(x)$ (this can be made through efficient algorithms), in order to find the value of d.

OTD3 Attack Strategy. The third attack strategy consists in considering the i-th row of the inverse of $\mathbf{G}'_{\leq k}$, that is:

$$\mathbf{R}_i = [\mathbf{Q}_i\mathbf{S}_{i,0}|\mathbf{Q}_i\mathbf{S}_{i,1}|\ldots|\mathbf{Q}_i\mathbf{S}_{i,n_0-2}] \tag{9}$$

and the linear code generated by

$$\mathbf{G}_{OTD3} = (\mathbf{Q}_i\mathbf{S}_{i,0})^{-1} \cdot \mathbf{R}_i = \left[\mathbf{I}|\mathbf{S}_{i,0}^{-1}\mathbf{S}_{i,1}|\ldots|\mathbf{S}_{i,0}^{-1}\mathbf{S}_{i,n_0-2}\right]. \tag{10}$$

Such code admits an alternative generator matrix in the following form:

$$\mathbf{G}'_{OTD3} = \mathbf{S}_{i,0}\mathbf{G}_{OTD3} = [\mathbf{S}_{i,0}|\mathbf{S}_{i,1}|\ldots|\mathbf{S}_{i,n_0-2}] \tag{11}$$

that coincides with a block row of matrix $\mathbf{S}$. Since matrix $\mathbf{S}$ has been chosen sparse, the code contains low weight codewords. Precisely, $\mathbf{G}'_{OTD3}$ has row weight equal to $m(n_0 - 1)$, that is very small compared to the code length.

Low weight codewords can be effectively searched through Stern's algorithm [15] and its variants [4]. Once having found matrix $\mathbf{S}$, a significant part of the secret key can be revealed by using (7). For the present choice of the system parameters, searching for low weight codewords in the code generated by $\mathbf{G}_{OTD3}$ would require, on average, 2^{32} binary operations.

3.2 First Variant of the Cryptosystem

The fundamental issue that validates the three OTD attack strategies relies in the fact that both $\mathbf{S}$ and $\mathbf{Q}$ are sparse and that matrix $\mathbf{Q}$ has block-diagonal form.

However, the three OTD attacks can be countered by adopting dense $\mathbf{S}$ matrices, without altering the remaining system parameters. For example, $\mathbf{S}$ could have row/column weight approximately equal to $k_0p/2$, with odd weight blocks along the main diagonal, and even weight blocks elsewhere, in order to assure non-singularity of $\mathbf{S}$, so that no further check is needed.

The adoption of dense $\mathbf{S}$ matrices prevents the eavesdropper from obtaining $\mathbf{Q}_i$ and $\mathbf{S}_{i,j}$, even knowing $\mathbf{Q}_i\mathbf{S}_{i,j}$. In this case, in fact, the probability that $g_{i,j}(x)$ has exactly m^2 non-null coefficients, and that its support contains that of at least one shift of $q_i(x)$ becomes extremely small. Furthermore, when $\mathbf{S}$ is dense, the code generated by $\mathbf{G}_{OTD3}$ does not contain any more low weight codewords, so all the three OTD attacks strategies are countered.

This modification has no effect on the number of errors to be corrected by the secret code, since the error spreading is only due to matrix $\mathbf{Q}$, that is kept sparse (with row/column weight m).

On the other hand, the choice of a dense $\mathbf{S}$ influences complexity of the decoding stage, that, however, can be reduced by resorting to efficient computation algorithms for circulant matrices. Complexity of the cryptosystem with dense $\mathbf{S}$ will be evaluated in Section 7.

As concerns matrix $\mathbf{Q}$, the OTD attacks demonstrate that the choice of the block-diagonal form is weak from the security viewpoint, so we avoid it in the revised versions of the cryptosystem. For example, an alternative choice would consist in obtaining $\mathbf{Q}$ from a matrix of $n_0 \times n_0 = 4 \times 4$ circulant blocks with weight 2, except those along the main diagonal, that have weight 1, and by permuting randomly its block rows and columns.

In this case, the inclusion of very low weight blocks in matrix $\mathbf{Q}$ could seem a flaw. However, the absence of the block-diagonal structure prevents from attacking each single block, and attacking a whole row or column would be too involved (it would require $p\binom{p}{2}^3 \approx 2^{81}$ attempts).

3.3 Second Variant of the Cryptosystem

In this second variant, we adopt the following set of parameters: $n_0 = 3$, $d_v = 13$ and $p = 8192$. The increased security level is achieved at the cost of a slightly decreased transmission rate, from 0.75 to 0.67, that, however, remains higher than in the original version. On the other hand, the reduction in the total number of circulant blocks permits to double their size without increasing the key length.

Both the private and the public codes, in this system, have dimension $k_0 p = 16384$ bit and length $n_0 p = 24576$. By means of numerical simulations, we have verified that such QC-LDPC codes are able to correct up to more than 470 errors per frame. For such reason, it has been possible to choose $t' = 40$ and $m = 11$ for this variant of the cryptosystem.

Matrix $\mathbf{Q}$ is formed by $n_0 \times n_0 = 3 \times 3$ circulant blocks with size p, and has row/column weight equal to $m = 11$. In this case, a possible choice consists in obtaining $\mathbf{Q}$ from a matrix of $n_0 \times n_0$ circulant blocks with weight 4, except those along the main diagonal, that have weight 3, and by permuting randomly its block rows and columns. In this case, attacking a whole row or column of $\mathbf{Q}$ would require $\binom{p}{4}^2\binom{p}{3} \approx 2^{131}$ attempts.

Matrix $\mathbf{S}$, instead, is formed by $k_0 \times k_0 = 2 \times 2$ circulant blocks with size p and it is dense, with row/column weight approximately equal to $k_0 p/2$. All its blocks have even row/column weight, except those along the main diagonal, that have odd weight, in order to allow non-singularity of the matrix.

3.4 Other Attacks

Having discussed above the OTD attack, in the following we list some of the other most important attacks with their corresponding work factor for the second cryptosystem variant, in order to assess its security. For a thorough description of all the attack techniques already considered, we refer the interested reader to [16]. where it is also proved that they are avoided for the parameters chosen in the previous version of the cryptosystem (and, hence, in the first variant here proposed).

Brute force attacks. Brute force attacks could be tempted by enumerating all possible secret matrices $\mathbf{H}$. However, the number of equivalent QC-LDPC codes with the proposed choice of the system parameters is $> 2^{386}$. Even considering a single circulant block in $\mathbf{H}$, the number of possible choices would be $> 2^{122}$.

Information set decoding attacks. Information set decoding attacks could be tempted through two different strategies. The first one consists in Lee and Brickell's method [3], that, however, would require 2^{94} operations.

Alternatively, the vector of intentional errors $\mathbf{e}$ could be searched as the lowest weight codeword in the extended code generated by $\mathbf{G}'' = \begin{bmatrix} \mathbf{G}' \\ \mathbf{x} \end{bmatrix}$. With the new choice of the parameters, however, this search would be very involved. By using the Stern algorithm, for example, it would require more than 2^{82} operations. For the first variant, instead, a similar search would require 2^{71} operations.

At the current stage of cryptanalysis, these attacks achieve the minimum work factor, that can be hence considered as the security level of each cryptosystem variant.

Attacks to the dual code. When the sparse structure of the private $\mathbf{H}$ is not sufficiently hidden, the eavesdropper could recover it by searching for low weight codewords in the dual of the public code, that admits $\mathbf{G}'$ has parity-check matrix. In this version of the cryptosystem, however, the dual of the public code does not contain low weight codewords, due to the effect of matrix $\mathbf{Q}$ on the rows of the private matrix $\mathbf{H}$.

In the present case, matrix $\mathbf{Q}$ has row/column weight 11, so the dual of the public code has codewords with weight $\leq n_0 \cdot d_v \cdot m = 429$. Due to the fact that the rows of $\mathbf{H}$ are sparse, it is highly probable that the minimum weight codewords in the dual of the public code have weight close to $n_0 \cdot d_v \cdot m$. However, even a lower weight would suffice to avoid attacks to the dual code. For example, if we suppose the existence of $p = 8192$ codewords with weight 150 in the dual of the public code (that has length $n = 24576$ and dimension $p = 8192$), searching for one of them through the Stern algorithm would require more than 2^{92} operations.

4 Fast Computations with Circulant Matrices

By exploiting the isomorphism described in Section 2, between the algebra of $p \times p$ binary circulant matrices and the ring of polynomials $\mathbb{R} = GF(2)[x]/(x^p + 1)$, computing the determinant of $\mathbf{S}$ and $\mathbf{Q}$ to check their invertibility, and computing the $k_0 \times n_0$ public matrix $\mathbf{G}' = \mathbf{S}^{-1}\mathbf{G}\mathbf{Q}^{-1}$, require only a few tens of operations in the ring $\mathbb{R}$. Anyway, the key generation process is not the critical one for efficiency, so we will focus complexity analysis on the vector-matrix products used for encryption and decryption.

4.1 Vector-Matrix Product

The isomorphism extends also to vector-matrix product. Let us suppose to have a vector $u = (u_0, u_1, \ldots, u_{p-1})$, and a circulant $p \times p$ matrix $\mathbf{A}$ mapped to the polynomial $a(x) \in \mathbb{R}$ by the isomorphism. The product $\mathbf{w} = \mathbf{u} \cdot \mathbf{A} = (w_0, w_1, \ldots, w_{p-1})$ will then be the vector whose components satisfy the equation $\sum_{i=0}^{p-1} w_i x^i \equiv (\sum_{i=0}^{p-1} u_i x^i) \cdot a(x) \pmod{x^p + 1}$. This vector-matrix product computation can be accelerated basically with two possible strategies: using fast polynomial multiplication algorithms based on evaluation-interpolation strategies, or using optimized vector-matrix product exploiting the Toeplitz structure.

To compare the methods, we will count the total number of bit operations needed. We will consider, for the naïve implementation, a number of operations given by the number of non-zero entries in the matrix. For the dense scenario we will consider that half of the entries are non-null. The starting value is hence $p^2/2$ operations. The two phases we will focus on are:

– The product $\mathbf{u} \cdot \mathbf{G}'$ used for encryption, where $\mathbf{u}$ is the message, seen as a vector of k_0 elements in $\mathbb{R}$, and $\mathbf{G}'$ is the public $k_0 \times n_0$ matrix with entries in $\mathbb{R}$. The cost with the naïve estimate is $p^2 k_0 n_0 / 2$ operations.
– The last step of decryption, that is, the product $\mathbf{u}' \cdot \mathbf{S} = \mathbf{u}$, where $\mathbf{u}'$ is again a k_0 vector in $\mathbb{R}$, and $\mathbf{S}$ is a $k_0 \times k_0$ invertible matrix. The naïve cost for this step is $p^2 k_0^2 / 2$.

5 Fast Polynomial Product

All the algorithms for fast polynomial multiplication are based on the same scheme: evaluation, point-wise multiplication, interpolation. The first strategy of this kind was proposed by Karatsuba [17] and then generalized by Toom and Cook [18,19]. We will call both of them Toom-s, with s the splitting order [20].

Other asymptotically faster algorithms exist for $GF(2)$; the most interesting ones are due to Cantor and Schönhage [21,22]. Another approach is the use of segmentation, also known as Kronecker-Schönhage's trick, but the threshold between Toom-Cook and all these methods is far above 100 000 bits [23].

5.1 General Toom-Cook Approach

The Toom-s algorithm for polynomial product requires five steps:

– **Splitting:** The two operands are represented by two polynomials (f and g) with s coefficients.
– **Evaluation:** f and g are evaluated in $2s - 1$ points.
– **Point-wise multiplication:** Computed evaluations are multiplied to obtain evaluations of the product, for example $(f \cdot g)(0) = f(0) \cdot g(0)$.
– **Interpolation:** Once the values of the product $f \cdot g$ in $2s - 1$ points are known, the coefficients are obtained via interpolation.
– **Recomposition:** The coefficients of the result are combined to obtain the product of the original operands.

Starting from [24], where the Toom-2,3,4 algorithms in $GF(2)$ where described in full details, we can estimate the cost of any one of the steps. Each product of two polynomials would require the cost for evaluation C_e counted twice, the cost for point-wise multiplication C_m, plus the cost for interpolation and recomposition C_i. Since we are dealing with vector-matrix products, where the matrix is fixed *a priori*, we can assume that all evaluations for the fixed operand are pre-computed.

Moreover, when multiplying a k_0 vector by a $k_0 \times n_0$ matrix, we can reduce the count to the strictly needed operations. We assume a pre-computation for all the evaluations of the matrix entries, and we need to evaluate the vector components only once: so we will count only k_0 evaluations. After the $n_0 k_0$ point-wise multiplications, we combine evaluated products $f_i g_i(\alpha), f_j g_j(\alpha), \ldots$ to obtain evaluations for the results like

$$(f_i g_i + f_j g_j + \cdots)(\alpha) \ ,$$

then we interpolate only the n_0 result polynomials: so we count only n_0 interpolations and recompositions.

5.2 Toom-2, also Known as Karatsuba

The Toom-2 algorithm splits the operands in two parts. Starting from two p-bits long polynomials we operate on $\lceil p/2 \rceil$-bits long parts.

Evaluation requires an addition of two parts. The 3 point-wise multiplication operates on parts, and gives doubled parts for results. Interpolation requires 5/2 additions on such doubled parts. Since an addition of two polynomials in $GF(2)[x]$ with degree d requires d operations, we can conclude that the use of Karatsuba to multiply a degree p polynomial by a fixed one requires:

- $\lceil p/2 \rceil$ operations for the evaluation,
- 3 multiplications of polynomials with degree $\lceil p/2 \rceil$,
- $5\lceil p/2 \rceil$ operations for the interpolation.

5.3 Cost of Exact Divisions

All the Toom-s with splitting order $s > 2$ require some exact divisions. We need to evaluate the cost of this kind of operation.

First of all, we highlight the fact that all the divisions needed for Toom-3 and Toom-4 in $GF(2)[x]$ are divisions by binomials of the form $x^w + 1$, with $w \in \{1, 2, 3\}$. Moreover, all the divisions in the Toom-Cook algorithm are exact divisions [24] and, therefore, can be computed in linear time [25]. When the divisor is in the very special form $x^w + 1$, exact division can be obtained *in-place* with the following simple code.

Input: the degree d, the vector $(a_0, \ldots, a_d)$ of coefficients of the polynomial $A = \sum_{i=0}^{d} a_i x^i$, the divisor in the form $D = x^w + 1$.

Output: the overwritten vector $(a_0, \ldots, a_{d-w})$ of coefficients of the new polynomial $A/D = \sum_{i=0}^{d-w} a_i x^i$.

Execution: for $i = 0 \ldots (d - w)$, $a_{i+w} \leftarrow a_{i+w} + a_i$.

So, a little bit less than d operations are required for any exact division needed in Toom-3 and Toom-4 on $GF(2)$. Since the division process cannot be parallelized as the addition does, we add an extra weight and we double the cost: we consider $2d$ bit-operation for each exact division. The given algorithm overwrites its input, so it should be slightly modified for general use; however, all the algorithms presented in [24], and proposed here, works with *in-place* operations. Other operations used in Toom-Cook are bit-shifts, but they can be avoided with word-alignment [23] in software implementation, and they have practically no cost in hardware implementations.

5.4 Toom-3, and Toom-4

From [24], we take the number of basic operations needed for the 3 and 4-way splitting. As usual we consider p-bits operands.

With Toom-3 we operate on $\lceil p/3 \rceil$-bits long parts. Evaluation requires 5 additions and 2 shifts per operand. The 5 point-wise multiplications involve $\lceil (p/3)+2 \rceil$-bits long parts, because of evaluations in x and $x+1$, which increase the degree. Interpolation operates on doubled parts, and requires 10 additions, 2 shifts and 2 divisions by $x+1$, plus 2 other additions for final recomposition; in total we have a cost of $10+2\cdot 2+2$ additions.

The Toom-3 product by a fixed operand hence requires:

- $\lceil p/3 \rceil \cdot 5$ bit operations for the evaluation,
- 5 multiplications of polynomials with degree $\lceil (p/3)+2 \rceil$,
- $\lceil (p/3)+2 \rceil \cdot 2 \cdot 17$ bit operations for the interpolation.

For Toom-4 we have $\lceil p/4 \rceil$-bits long parts. Evaluation requires 15 additions and 9 shift per operand. The 7 point-wise multiplications involve $\lceil (p/4)+3 \rceil$-bits long parts. Interpolation operates on doubled parts, and requires 29 additions, 16 shifts and 4 divisions by x^2+1 and x^3+1, plus 3 other additions for final recomposition; in total we have the cost of $29+4\cdot 2+3$ additions.

The Toom-4 product by a fixed operand hence requires:

- $\lceil p/4 \rceil \cdot 15$ bit operations for the evaluation,
- 7 multiplications of polynomials with degree $\lceil (p/4)+3 \rceil$,
- $\lceil (p/4)+3 \rceil \cdot 2 \cdot 40$ bit operations for the interpolation.

All Toom-s methods can be used recursively and have asymptotic complexity $O(p^{\log_s(2s-1)})$, but the bigger the splitting order, the heavier the overhead of evaluation/interpolation.

5.5 Numerical Examples

Let us fix the following choice for the system parameters: $p = 8192, k_0 = 2, n_0 = 3$, that are the choices adopted in the second variant of the cryptosystem (see Section 3.3).

The use of 2 recursions of Toom-4, 1 of Toom-3 and 4 of Toom-2 reduces one p-bit multiplication to $7^2 \cdot 5 \cdot 3^4 = 19\,845$ 11-bits sized sub-products, each one with a cost of $11^2/2$ operations. We have an overhead of 301 655 operations for evaluations and 1 617 574 for interpolation. The total cost of computing $\mathbf{u} \cdot \mathbf{G}'$ is then $301\,655 \cdot k_0 + 1\,617\,574 \cdot n_0 + 7^2 \cdot 5 \cdot 3^4 \cdot 11^2 \cdot k_0 \cdot n_0/2 + n_0 \cdot p = 12\,684\,343$ bit-operations, included the final reduction modulo $x^p + 1$. This count gives a far smaller result than the naïve technique, that requires $k_0 \cdot n_0 \cdot p^2/2 = 201\,326\,592$ operations. With the same approach, we can obtain the cost of computing $\mathbf{u}' \cdot \mathbf{S} = \mathbf{u}$. 8 657 332 operations using Toom-Cook, versus 134 217 728 with a naïve implementation.

With parameters $p = 4096, k_0 = 3, n_0 = 4$, that are very similar to the choices adopted for the proposal in [9], we assume 3 recursions of Toom-4 and 3 of Toom-2, that result in a number of bit operations for $\mathbf{u} \cdot \mathbf{G}'$ of 8 074 444 versus 100 663 296 with the naïve approach. While for the decryption step $\mathbf{u}' \cdot \mathbf{S} = \mathbf{u}$ we obtain 6 166 127 bit operations versus 75 497 472.

6 Vector-Toeplitz Convolution

Another approach to speed-up polynomial products in cryptography is the Winograd convolution [26]. It is very similar to Karatsuba's multiplication, but it has a smaller overhead. We shortly recall it. Given an even sized $2d \times 2d$ Toeplitz matrix $\mathbf{T}$, we can *factorize* it as follows:

$$\begin{pmatrix} \mathbf{T}_0 & \mathbf{T}_1 \\ \mathbf{T}_2 & \mathbf{T}_0 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{T}_1 - \mathbf{T}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_2 - \mathbf{T}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{T}_0 \end{pmatrix} \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} \\ \mathbf{I} & \mathbf{I} \end{pmatrix} ,$$

where $\mathbf{I}$ is the $d \times d$ identity matrix, and $\mathbf{T}_0, \mathbf{T}_1, \mathbf{T}_2$ are themselves $d \times d$ Toeplitz matrices, as also $\mathbf{T}_1 - \mathbf{T}_0$ and $\mathbf{T}_2 - \mathbf{T}_0$. It follows that the vector-matrix product $(\mathbf{V}_0, \mathbf{V}_1) \cdot \mathbf{T}$ can be computed with three steps:

- the addition $\mathbf{V}_0 + \mathbf{V}_1$,
- 3 vector-matrix sub-products by $d \times d$ Toeplitz matrices,
- 2 more additions to obtain the result.

Since circulant matrices are also Toeplitz and the proposed size p is a power of 2, this optimization can be used for as many recursions as needed for our computations. Asymptotic complexity is exactly the same for Toom-2 and this approach, but if we analyze again the pre- and post-computation, we obtain:

- $p/2$ operations for the "evaluation",
- 3 multiplications with dimension $p/2$,
- $p/2$ operations for the "interpolation".

6.1 Numerical Examples

This method cannot be mixed with Toom-Cook, and its main advantage is that it is much easier to implement in software.

If we consider again $p = 8192, k_0 = 2, n_0 = 3$, the use of 11 recursions of Winograd's method leads to 14 106 224 operations for encryption versus 12 684 343 obtained with Toom. The decryption step $\mathbf{u}' \cdot \mathbf{S} = \mathbf{u}$ can be completed in 9 871 080 operations, around 15% more than with the polynomial strategy, but with a far simpler source code. With smaller parameters, for example $p = 4096, k_0 = 3, n_0 = 4$, the difference is smaller: 11 recursions used for encryption give 8 103 706 operations, only 1% more than the cost computed using Toom-Cook.

On the other hand, the use of polynomials gives greater flexibility. For example, with the odd parameter $p = 5555, k_0 = 2, n_0 = 3$, 2 recursions of Toom-4 and 5 of Toom-2 give a cost for encryption of 7 310 809 bit operations, a 12× speed-up with respect to the naïve approach. For the same parameters, Winograd's trick is not applicable at all, because p is odd.

7 Cryptosystem Complexity Assessment

In this section we evaluate the encryption and decryption complexity of the proposed cryptosystem, by considering the usage of efficient computation algorithms suitable for the particular structure of the matrices involved.

Encryption complexity is due to multiplication of the cleartext by the code generator matrix and to addition of intentional errors. It can be expressed as follows:

$$C_{enc} = C_{mul}\left(\mathbf{u} \cdot \mathbf{G}'\right) + n \tag{12}$$

where $C_{mul}\left(\mathbf{u} \cdot \mathbf{G}'\right)$ represents the number of operations needed for calculating the product $\mathbf{u} \cdot \mathbf{G}'$ and n binary operations are considered for the addition of vector $\mathbf{e}$.

The decryption complexity, instead, can be divided into three parts:

$$C_{dec} = C_{mul}\left(\mathbf{x} \cdot \mathbf{Q}\right) + C_{SPA} + C_{mul}\left(\mathbf{u}' \cdot \mathbf{S}\right) \tag{13}$$

where $C_{mul}\left(\mathbf{x} \cdot \mathbf{Q}\right)$ and $C_{mul}\left(\mathbf{u}' \cdot \mathbf{S}\right)$ represent the number of operations needed for computing $\mathbf{x} \cdot \mathbf{Q}$ and $\mathbf{u}' \cdot \mathbf{S}$, respectively, while C_{SPA} is the number of operations required for LDPC decoding through the sum-product algorithm. In expressions (12) and (13), $C_{mul}\left(\mathbf{u} \cdot \mathbf{G}'\right)$ and $C_{mul}\left(\mathbf{u}' \cdot \mathbf{S}\right)$ involve multiplication by dense matrices, so we can resort to efficient algorithms, like the Toom-Cook method described in the previous section. $C_{mul}\left(\mathbf{x} \cdot \mathbf{Q}\right)$, instead, expresses the number of operations needed to perform the product of a $1 \times n$ vector by a sparse $n \times n$ matrix ($\mathbf{Q}$, with row/column weight equal to m). For this reason, we resort to the naïve implementation, that has the lowest complexity $C_{mul}\left(\mathbf{x} \cdot \mathbf{Q}\right) = n \cdot m$.

For the decoding complexity, the following expression can be adopted [16]:

$$C_{SPA} = I_{ave} \cdot n \left[q\left(8d_v + 12R - 11\right) + d_v\right] \tag{14}$$

where I_{ave} is the average number of decoding iterations, q is the number of quantization bits used inside the decoder and $R = k_0/n_0$ is the code rate. Numerical simulations have permitted to verify that, for the codes involved in the first cryptosystem implementation, assuming $q = 6$ and $t = 190$, it is $I_{ave} \simeq 5$.

For the codes used in the new variant, instead, assuming $q = 6$ and $t = 470$, it is $I_{ave} \simeq 9$. These values of I_{ave} are further reduced for smaller t.

As concerns the public key length, the proposed cryptosystem uses, as the public key, a generator matrix, $\mathbf{G}'$, formed by $k_0 \times n_0$ circulant blocks with size p. Therefore, it can be completely described by $k_0 \cdot n_0 \cdot p$ bits.

	McEliece (original)	Niederreiter	RSA	QC-LDPC McEliece 1	QC-LDPC McEliece 2
Key Size (bytes)	67072	32750	256	6144	6144
Information Bits	524	276	1024	12288	16384
Transmission Rate	0.5117	0.5681	1	0.75	0.6667
Enc Ops per bit	514	50	2402	658	776
Dec Ops per bit	5140	7863	738 112	4678	8901

Table 1. Comparison between the proposed versions of QC-LDPC based McEliece cryptosystem and other schemes

Table 1 reports the characteristics of the two proposed variants of McEliece cryptosystem based on QC-LDPC codes, both secure against the known attacks. The first variant (noted as QC-LDPC McEliece 1) adopts the choice of the system parameters we have already proposed in [9], with the only difference of $p = 4096$ instead of 4032. We have considered p coincident with a power of two also in this version because, for such values, a circulant matrix with odd row/column weight is always non-singular, as shown in the Appendix A. The choice of $p = 4096$ instead of 4032, however, has no effect on the system security. In the second variant (noted as QC-LDPC McEliece 2), the system parameters have been changed in order to increase the system security level. From the sake of comparison, also the original McEliece, the Niederreiter and the RSA cryptosystems are considered. The key length for the Niederreiter cryptosystem coincides with the number of bits in the non-systematic part of matrix $\mathbf{H}$.

From the security viewpoint, these systems are not equivalent: the first proposal exhibits a security level of 2^{71} binary operations, while the second one exceeds the threshold of 2^{80} binary operations, that is currently considered as an up to date technology limit. The first three solutions are instead assumed with their standard parameters [4]. In such case, the McEliece and Niederreiter cryptosystems are not able to reach a similar security level; however, more secure versions would yield increased complexity.

It results from the table that both the proposed variants of McEliece cryptosystem based on QC-LDPC codes represent a trade-off between the original McEliece cryptosystem (and its Niederreiter version) and other cryptosystems, like RSA. In fact, they represent an advance in overcoming the drawbacks of the original McEliece cryptosystem: they have very smaller public keys and increased transmission rate. With respect to RSA, the proposed cryptosystems have the advantage of very lower complexity, that is only slightly increased with respect to the original McEliece version (that, moreover, has a lower security level).

8 Conclusions

We have elaborated on an implementation of the McEliece cryptosystem based on QC-LDPC codes we have recently proposed for overcoming the main drawbacks of its original version, that has been recently discovered to be subject to dangerous attacks. We have described how these attacks exploit the sparse character of some constituent matrices, together with their diagonal form, and we have proposed two new variants of the cryptosystem that do not allow the application of such attack techniques. As typical in cryptography, this does not exclude that further attacks might be conceived in the future. So, an effort should be made for getting a coding based cryptographic construction with a supporting proof of security, similar to what done in the related area of lattice based cryptography [27]. For the time being, based on our knowledge, we can say that possible progress in cryptanalysis of the proposed system will require the definition of substantially new strategies.

We have also reported complexity estimates based on the Toom-Cook method for polynomials in $GF(2)[x]$. They can be partially extended to other cryptosystems, such as NTRU, where polynomials modulo $x^n \pm 1$ are used, and ECC on $GF(2^n)$. For both these systems, the use of Karatsuba's and Winograd's fast convolution were proposed [28,29], and the Toom-Cook method could be applied as well, even if its effect should be not as impressive as in the proposed system.

The promising results obtained with Toom-Cook have their main reason in the use of matrices. The additional cost of fast multiplication methods is quite big, that is the reason why they are effective only for big operands. Numerical examples given in Section 5.5 show that, for deep recursion, more than half the cost of a single product comes from evaluation and interpolation. This cost, however, is reduced in the proposed cryptosystem, because only a few evaluations and interpolations are needed for a set of multiplications, so that deepest recursion and biggest saving are possible.

The application of the Toom-Cook method permits to reduce the encryption and decryption complexity of the proposed cryptosystems, that, furthermore, are able to overcome the main drawbacks of the original McEliece cryptosystem in terms of key size and transmission rate. For these reasons, they can be seen as a valuable trade-off between the original McEliece cryptosystem and other widespread solutions, like RSA.

References

1. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. DSN Progress Report (1978) 114–116
2. Berlekamp, E., McEliece, R., van Tilborg, H.: On the inherent intractability of certain coding problems. IEEE Trans. Inform. Theory **24** (May 1978) 384–386
3. Lee, P., Brickell, E.: An observation on the security of McEliece's public-key cryptosystem. In Günther, C.G., eds.: EUROCRYPT 1988. Volume 330 of LNCS., Springer (1988) 275–280

4. Canteaut, A., Chabaud, F.: A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. IEEE Trans. Inform. Theory **44** (January 1998) 367–378
5. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Probl. Contr. and Inform. Theory **15** (1986) 159–166
6. Li, Y.X., Deng, R., Wang, X.M.: On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. IEEE Trans. Inform. Theory **40**(1) (January 1994) 271–273
7. Riek, J.: Observations on the application of error correcting codes to public key encryption. In: Proc. IEEE International Carnahan Conference on Security Technology. Crime Countermeasures, Lexington, KY, USA (October 1990) 15–18
8. Richardson, T., Urbanke, R.: The capacity of low-density parity-check codes under message-passing decoding. IEEE Trans. Inform. Theory **47** (2001) 599–618
9. Baldi, M., Chiaraluce, F.: Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In: Proc. IEEE ISIT 2007, Nice, France (June 2007) 2591–2595
10. Monico, C., Rosenthal, J., Shokrollahi, A.: Using low density parity check codes in the McEliece cryptosystem. In: Proc. IEEE ISIT 2000, Sorrento, Italy (June 2000) 215
11. Otmani, A., Tillich, J.P., Dallot, L.: Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. In: Proc. First International Conference on Symbolic Computation and Cryptography (SCC 2008), Beijing, China (April 2008)
12. Gaborit, P.: Shorter keys for code based cryptography. In: Proc. Int. Workshop on Coding and Cryptography WCC 2005, Bergen, Norway (March 2005) 81–90
13. Richardson, T., Urbanke, R.: Efficient encoding of low-density parity-check codes. IEEE Trans. Inform. Theory **47** (February 2001) 638–656
14. Neal, R.M.: Faster encoding for low-density parity check codes using sparse matrix methods (1999) `http://www.cs.toronto.edu/~radford/ftp/ima-part1.pdf`.
15. Stern, J.: A method for finding codewords of small weight. In Cohen, G., Wolfmann, J., eds.: Coding Theory and Applications. Volume 388 of LNCS., Springer (1989) 106–113
16. Baldi, M., Chiaraluce, F.: LDPC Codes in the McEliece Cryptosystem (September 2007) `http://arxiv.org/abs/0710.0142`.
17. Karatsuba, A.A., Ofman, Y.: Multiplication of multidigit numbers on automata. Soviet Physics Doklady **7** (1963) 595–596
18. Toom, A.L.: The complexity of a scheme of functional elements realizing the multiplication of integers. Soviet Mathematics Doklady **3** (1963) 714–716
19. Cook, S.A.: On the minimum computation time of functions. PhD thesis, Dept. of Mathematics, Harvard University (1966)
20. Bodrato, M., Zanoni, A.: Integer and polynomial multiplication: Towards optimal Toom-Cook matrices. In Brown, C.W., ed.: Proceedings of the ISSAC 2007 Conference, New York, NY, USA, ACM press (July 2007) 17–24
21. Cantor, D.G.: On arithmetical algorithms over finite fields. Journal of Combinatorial Theory **A 50** (1989) 285–300
22. Schönhage, A.: Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. Acta Informatica **7** (1977) 395–398
23. P. Brent, R., Gaudry, P., Thomé, E., Zimmermann, P.: Faster multiplication in GF(2)[x]. In van der Poorten, A.J., Stein, A., eds.: Proceedings of the Eighth Algorithmic Number Theory Symposium (ANTS-VIII). Volume 5011 of LNCS., Springer (May 2008) 153–166

24. Bodrato, M.: Towards optimal Toom-Cook multiplication for univariate and multivariate polynomials in characteristic 2 and 0. In Carlet, C., Sunar, B., eds.: WAIFI 2007 proceedings. Volume 4547 of LNCS., Springer (June 2007) 116–133
25. Jebelean, T.: An algorithm for exact division. Journal of Symbolic Computation **15** (1993) 169–180
26. Winograd, S.: Arithmetic Complexity of Computations. Volume 33 of CBMS-NSF Regional Conference Series in Mathematics. SIAM (1980)
27. Micciancio, D.: Generalized compact knapsacks, cyclic lattices and efficient one-way functions. Computational Complexity **16**(4) (2007) 365–411
28. Silverman, J.H.: High-speed multiplication of (truncated) polynomials. Technical Report 10, NTRU CryptoLab (January 1999)
29. Weimerskirch, A., Stebila, D., Shantz, S.C.: Generic GF(2) arithmetic in software and its application to ECC. In Safavi-Naini, R., Seberry, J., eds.: ACISP. Volume 2727 of LNCS., Springer (2003) 79–92

A Matrix Inversion in $GF(2)[x]/(x^p + 1)$

For whatever choice of p, we work on the polynomial ring $\mathbb{R} = GF(2)[x]/(x^p+1)$. In any case $x^p + 1$ is divisible by $x + 1$, because we are working in characteristic 2, so that the ring is not a field: it has zero divisors and non invertible elements.

If the chosen p is a power of two ($p = 2^a$), then $x+1$ is the only prime factor of $x^p + 1 \equiv (x + 1)^p$ and, in such a case, it is very easy to check if an element in $\mathbb{R}$ is not invertible. In fact, it suffices checking if 1 is a root, or, equivalently, if the number of non-zero (1) coefficients is even.

If p is not a power of two, the invertibility check becomes more involved. Obviously general theorems are still valid, so that we can say that a generic element $f \in \mathbb{R}$ is invertible if and only if it is coprime with $x^p + 1$, and a matrix is invertible if and only if its determinant is invertible. But invertible matrices can exist that only contain non invertible entries.

For example, with $p = 3$, we have $x^3 + 1 \equiv (x + 1) \cdot (x^2 + x + 1) \pmod 2$. Neither $x+1$ nor x^2+x+1 are invertible, but the following matrix is non-singular:

$$\begin{pmatrix} x+1 & x^2+x+1 \\ x^2+x+1 & x+1 \end{pmatrix} \cdot \begin{pmatrix} x^2+1 & x^2+x+1 \\ x^2+x+1 & x^2+1 \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \in GF(2)[x]/(x^3+1).$$

So one needs a clever algorithm to compute the inverses of the matrices, either by computing the inverse on any sub-ring $GF(2)[x]/d^\alpha$ where $d^\alpha | x^p + 1$, then combining the results with the Chinese Remainder Theorem, or by a modified version of Gaussian inversion, exploiting Bezout's identity to obtain a pivot on columns without invertible elements.

THIS PAGE IS NOT PART OF THE ARTICLE

B BibTEX entry

```
@InProceedings{Bodrato:SCN2008,
  author = {Baldi, Marco and Bodrato, Marco and Chiaraluce, Franco},
  title =  {A New Analysis of the {McEliece} Cryptosystem
            based on {QC-LDPC} Codes},
  pages =  {246--262},
  url =    {http://bodrato.it/papers/#SCN2008},
  crossref =  {SCN2008},

  year =   {2008},
  booktitle = {{SCN 2008} Proceedings},
  editor = {Ostrovsky, Rafail and De Prisco, Roberto and Visconti, Ivan },
  volume = {5229},
  series = {LNCS},
  month =  {September},
  publisher = {Springer}
}

@Proceedings{SCN2008,
  title = {Security and Cryptography for Networks,
            6th International Conference},
  year =  {2008},
  location =  {Amalfi, Italy},
  booktitle = {{SCN2008} Proceedings},
  editor = {Ostrovsky, Rafail and De Prisco, Roberto and Visconti, Ivan },
  volume = {5229},
  series = {Lecture {N}otes in {C}omputer {S}cience},
  month =  {September},
  publisher = {Springer}
}
```