On McEliece type cryptosystems using self-dual codes with large minimum weight

Luca Mariot, Stjepan Picek, and Radinka Yorgova

Delft University of Technology, Delft, The Netherlands
L.Mariot@tudelft.nl, S.Picek@tudelft.nl, r.a.yorgova@student.tudelft.nl

Abstract. One of the finalists in the NIST post-quantum cryptography competition is the Classic McEliece cryptosystem. Unfortunately, its public key size represents a practical limitation. One option to address this problem is to use different families of error-correcting codes. Most of such attempts failed as those cryptosystems were proved not secure. In this paper, we propose a McEliece type cryptosystem using high minimum distance self-dual codes and punctured codes derived from them. To the best of our knowledge, such codes have not been implemented in a code-based cryptosystem until now. For the 80-bit security case, we construct an optimal self-dual code of length 1 064, which, as far as we are aware, was not presented before. Compared to the original McEliece cryptosystem, this allows us to reduce the key size by about 38.5%.

Keywords: Post-quantum cryptography · McEliece cryptosystem · Self-dual codes.

1 Introduction

The process initiated by NIST to standardize one or more quantum-resistant public-key cryptographic algorithms is ongoing, and currently, in round 3 [1] [37]. One of the four finalists for the public-key encryption and key-establishment algorithms standard is the Classic McEliece cryptosystem. This fact indicates that after a long time of research on the original encryption scheme [31], it remains one of the most proven secure public-key cryptosystems.

Still, there is a major drawback, namely the size of its public key. This is a practical limitation for broad use in the current communication systems. For comparison, for the 128 bits security level of the McEliece cryptosystem, the size of its public key is around 187.69 Kb [8], whereas the public key of RSA for the same bit security is 3 Kb (or equivalently, 3 072 bits) [36, Table 2].

A significant number of studies aim to minimize the key size of the McEliece cryptosystem by using different families of error-correcting codes. Most of the proposed cryptosystems in the short term have been proven not secure. One common characteristic of these systems, in contrast with the original one, is that they use codes with a low error-correction capability [22,3,34].

[1] As of June 2021.

This paper proposes a McEliece type cryptosystem using codes with error-correction capability higher than the capability of the codes adopted until now. As such, this work can be seen as a study on the trade-off between the error-correction capability and the size of the public key. More specifically, we use high minimum distance self-dual codes and punctured codes derived from them. To the best of our knowledge, such codes have not been implemented in a code-based cryptosystem until now. The reason is most likely twofold: first, self-dual codes are known up to length 130, which is too small for current security requirements. Second, there is no fast hard-decision decoding algorithm for such codes, an exception being the extended Golay code [40].

Our Contributions. This work studies the trade-off between the error-correcting capability and the size of the implemented code in a McEliece type cryptosystem. We use high minimum distance binary self-dual codes and their punctured codes with a high error-correction capability. We call this encryption scheme a *McEliece type cryptosystem* as it uses a different type of codes from the binary Goppa codes as used in McEliece's proposal.

A small example of the cryptosystem using a code obtained from an optimal self-dual code of length 104 is implemented in SageMath. For the decryption process, an appropriate decoding algorithm is adapted and implemented. Security analysis shows that the resulting cryptosystem has at least a 22-bit security level using a key of size 0.3251 Kb, whereas the key of the original McEliece cryptosystem with the same bit security level is at least 0.462 Kb, i.e., our example reduces the key size by about 30%.

Next, we determine the parameters of a putative optimal self-dual code, which, if implemented in a McEliece type cryptosystem, would provide a classic security level of 80, 128, and 256 (quantum 67, 101, and 183) bits, respectively. Moreover, for the 80-bit security case, we construct an optimal self-dual code of length 1 064. To the best of our knowledge, such a code is presented for the first time. We further derive a punctured code from this example to be used as a private key for decryption.

Our theoretical analysis estimates that the security level of the complete system is 80 and 67 bits against classical and quantum attacks, respectively. The size of the resulting public key is 276.39 Kb, whereas the best-known example of a binary Goppa code providing the same bit security level in the original McEliece cryptosystem is 449.85 Kb [8]. Therefore, in this case, we achieve a reduction of the key size around 38.5%. *The results on the 80-bit security case suggest that self-dual codes can be used in practice in a McEliece type cryptosystem to reduce the key size for the same security level.*

2 Background

Let $\mathbb{F}_2^n$ be the n-dimensional vector space over the binary field $\mathbb{F}_2$, and let $\mathcal{D} \subseteq \mathbb{F}_2^n$ be a k-dimensional subspace of $\mathbb{F}_2^n$. The *Hamming distance* between two vectors in $\mathbb{F}_2^n$ is the number of coordinates where they differ, while the *Hamming weight* (or only *weight*) $wt(v)$ of a vector $v \in \mathbb{F}_2^n$ is the number of the nonzero coordinates of v. A subspace $\mathcal{D}$ of $\mathbb{F}_2^n$ is called a binary linear code $[n, k, d]$ where d is the minimum Hamming distance between any pair of vectors (also called *codewords*) of $\mathcal{D}$. Equivalently, d is the minimum weight among all nonzero codewords of $\mathcal{D}$. The inner product in $\mathbb{F}_2^n$ is given by $\langle u, v\rangle = u_1v_1 + u_2v_2 + \cdots + u_nv_n$ for $u,\ v \in {\mathbb{F}_2}^n$, and u and v are *orthogonal* if such product is equal to 0. Then, $\mathcal{D}^\perp = \{v \in \mathbb{F}_2^n :\ \langle u, v\rangle = 0,\ \forall u \in \mathcal{D}\}$ is the orthogonal of the code $\mathcal{D}$.

The code $\mathcal{D}$ is called *self-orthogonal* if $\mathcal{D} \subset \mathcal{D}^\perp$, and *self-dual* if $\mathcal{D} = \mathcal{D}^\perp$, $\mathcal{D}$. It is known that the weight of any codeword of a binary self-dual code is even. If an error-correcting code is a linear $[n, k, d]$ code then it can correct $t \leq (d-1)/2$ errors. Let C be a linear code and C_i the set of all words of C without the i-th coordinate. Then, C_i is the punctured code of C on the i-th position.

2.1 McEliece Cryptosystem

The McEliece Cryptosystem is the first code-based cryptosystem proposed by Robert McEliece in 1978 [31]. The original cryptosystem uses a binary $[1\,024, 524]$ code with an error-correcting capability of 50 errors. The steps of the encryption scheme are as follows:

1. *Define the system parameters:* k - the length of the message block, n - the length of the ciphertext, t - the number of the intentionally added errors (equal to the error-correcting capability of the implemented linear code).
2. *Key generation:* define: G - a generating matrix of an $[n, k, 2t + 1]$ code for which there is a fast decoding algorithm; P - a random $n \times n$ permutation matrix and S - a random dense $k \times k$ non-singular matrix and, compute $G' = SGP$, S^{-1} and P^{-1} - the inverse of P and S. Note that G' generates a linear code with the same n, k and t. Then, (G', t) - *Public key*, (G, P, S) or (Dec_G, P, S) - *Private key*, where Dec_G is the fast decoding algorithm.
3. *Encryption:* split the data for encryption into k-bit blocks. Then each block m is encrypted as $r = G'm + e$, where e is a random vector of length n and weight t.
4. *Decryption:* The received vector r is decrypted as follows:
   (a) Compute $r' = rP^{-1}$, which is $mSG + eP^{-1}$.
   (b) Decode r' into a codeword c' using the efficient decoding algorithm for the code with generator matrix G, $c' = mSG$.
   (c) Compute c such that $cG = c'$ (If G is in a systematic form, then c is the first k bits of c').
   (d) Compute $m = cS^{-1}$.

The scheme above can be applied with any linear code for which a fast decoding algorithm is known. In particular, the original system in [31] employs a binary $[1\,024, 524, 101]$ Goppa code.

2.2 Cryptanalysis

As with any other public encryption scheme, the McEliece cryptosystem gives the following information to the attacker: the encryption parameters, the encryption and decryption algorithms, and the public key. Hence, the adversary can also select any plaintext and compute the corresponding ciphertext.

Concerning the adversary goals (total break, partial break, and distinguishing break), there are three main categories of attacks:

- *Key-recovery attack*: the attacker deduces the private key.
- *Message-recovery attack*: the attacker obtains a part or complete plaintext corresponding to a ciphertext without knowing the private key.
- *Distinguishing attack*: the attacker can distinguish the cipher from a random message without knowledge about the private key.

Next, we consider a few of the known attacks towards the McEliece encryption scheme. For each attack, we evaluate *the probability of success* or the inverse problem of evaluating the average number of attempts of the attack until the attacker achieves its target.

For algorithmic attacks *the security level* of a system is defined as a minimum work factor. The *work factor* is the average number of elementary (binary) operations needed to perform a successful attack [2, p.72].

In the following sections, we describe the main attacks published in the relevant literature, assuming that a McEliece cryptosystem is defined by a private key (G, P, S), where G is a generator $k \times n$ matrix of a binary $[n, k, 2t+1]$ code, P is a random $n \times n$ permutation matrix, S is a random dense $k \times k$ non-singular matrix, and a public key (G', t) where $G' = SGP$. Further, we assume that the attacker has access to a ciphertext c produced by the encryption scheme. Thus, we start by first recalling the components over which brute-force attacks can be mounted. Then, we describe the basic ISD attack and its work factor, along with some of its improved versions, particularly Stern's ISD attack.

Brute-force Attacks. A brute-force attack can be mounted towards different components of the encryption system:

- *Towards the message*: the attacker takes a random message m_1 of length k, encrypts it to $c_1 = m_1 \cdot G'$, and computes the difference $e_1 = c - c_1$. If the difference e_1 has weight $\leq t$, then the plaintext corresponding to the ciphertext c is exactly m_1 and the attack succeeds. Then the probability of success is $1/2^k$ since the number of all possible messages of length k is 2^k.
- *Towards the coset leaders of the code generated by G'*: the attacker computes the syndrome of all coset leaders. The coset leader with syndrome equal to the syndrome of the ciphertext c is the error vector. Knowing the error vector, one can compute the codeword and then the message. The number of the coset leaders is $|\mathbb{F}_2^n|/|C'| = 2^{n-k}$. Therefore, the work factor of this attack is at least 2^{n-k}.
- *Towards the error-vector*: the attacker searches among the vectors e of length n and weight t such that the syndrome of e is equal to the syndrome of the received vector c (the ciphertext). Thus, it is a search on e such that $S(e) = e \cdot H^T$ equals $S(c)$, where H represents the parity-check matrix corresponding to G'. This problem is equivalent to the problem of finding a linear combination of t columns of H, which results in a column vector with

weight $S(c)$. Since there are $\binom{n}{t}$ possible choices for the vector e, then the work factor of the brute force attack towards the error vector is $\binom{n}{t}$.

Information Set Decoding Attacks (ISD). The Information Set Decoding (ISD) technique was introduced by Prange in 1962 [41] as an efficient decoding method for cyclic codes. Several works (e.g., [28,39,25]) considered increasingly improved versions of the ISD decoding algorithm to attack the original McEliece cryptosystem described in [31].

An *information set* for a $[n, k]$ code $\mathcal{C}$ is any subset $A = \{i_1, \cdots, i_k\}$ of k coordinates such that, for any given set of values $b_i \in \mathbb{F}_2$, with $i \in A$, there is a unique codeword $c \in C$. The information set thus consists of any k indices such that the corresponding k columns of a generator matrix of $\mathcal{C}$ have rank k.

Let $v = mG' + e$, where G' is a generator matrix of an $[n, k, 2t + 1]$ code $\mathcal{C}$ and e is an error vector of weight t. Let A be an information set of k coordinates such that all entries of the error vector indexed by A are 0. In summary, the algorithm for the ISD attack works as follows:

1. Choose k out of n indices for the information set. These k columns of G' are permuted to the first k positions, which is $G'P = [A_k|A_{n-k}]$, where A_k are the chosen k columns and A_{n-k} is the rest of G';
2. Transform the matrix $[A_k|A_{n-k}]$ in systematic form, which takes $\mathcal{O}(k^3)$ operations [31], since it entails solving k linear equations in k unknowns. This is equivalent to transforming $G'P$ into $[I_k|A'_{n-k}] = UG'P$, where U is the transformation matrix;
3. Compute m as the multiplication of v by the inverse matrix G_S^{-1}. Then $e = v - mG'$. If $wt(e) = t$, then m is the encrypted message. The possibilities for the error vector e to have 0 coordinates in the information set are k out of $n - t$ coordinates, i.e., $\binom{n-t}{k}$;
4. Estimate how many of the choices for k out of n columns have rank k of the generator matrices of the family of $[n, k, 2t + 1]_2$ codes. In the original code-based cryptosystem, Goppa codes were used and for these codes, around 29% of the choices of k columns are invertible.

Therefore, the work factor for the ISD attack is $\frac{k^3\binom{n}{k}}{\beta\binom{n-t}{k}}$, where β is the proportion of the invertible k columns out of n for the generator matrices of the family of $[n, k, 2t + 1]$ codes. Note that β depends on the specific family.

Stern's ISD Attack. Stern [47] proposed a refinement of the ISD attack, which is based on the following result:

Lemma 1. *[2, p.76] The $(n, k + 1)$ linear code generated by*

$$G'' = \begin{pmatrix} G' \\ x \end{pmatrix} = \begin{pmatrix} G' \\ u \cdot G' + e \end{pmatrix} \quad . \tag{1}$$

has only one minimum weight codeword, which coincides with e.

The idea behind the attack is to use the extended code generated by G'' and find the corresponding unique codeword e of weight t. Stern's algorithm is probabilistic, using two input parameters p and l together with the parity check matrix of the extended code.

The work factor of one iteration of the attack is $B = f_1 + f_2 + f_3$, where [47]:

$$f_1 = \frac{1}{2}(n-k)^3 + k(n-k)^2 \ , \tag{2}$$

$$f_2 = 2pl\binom{k/2}{p} \ , \tag{3}$$

$$f_3 = 2p(n-k)\frac{\binom{k/2}{p}^2}{2^l} \ . \tag{4}$$

The total work factor of the attack is $\frac{B}{P_t}$, where P_t is the probability of finding a codeword of weight t in one iteration. In particular, P_t is estimated in [47] as:

$$P_t = \frac{\binom{t}{2p}\binom{n-t}{k-2p}}{\binom{n}{k}} \cdot \frac{\binom{2p}{p}}{4^p} \cdot \frac{\binom{n-k-t+2p}{l}}{\binom{n-k}{l}} \ . \tag{5}$$

Quantum Basic Information Set Decoding Attack. Let $v = mG + e$, G and e be defined as before. We give the Basis Quantum Information Set Decoding function in Algorithm 1.

Algorithm 1: Basis Quantum Information Set Decoding function

---

1 Choose k coordinates $S = \{i_1, i_2, ..., i_k\}$ and form the matrix G_S.
  If $det(G_S) \neq 0$ **then** find ${G_S}^{-1}$ **else**, giving up
2 Compute $(v_{i_1}, v_{i_2}, ..., v_{i_k}) \cdot {G_S}^{-1} = m$, $m \in \mathbb{F}_2^k$
3 Compute $mG \in \mathbb{F}_2^n$
4 Compute $e = v - mG$. If $wt(e) \neq t$ then giving up
5 Returns 0.

Regarding [7], searching randomly a root of the function in Algorithm 1 can succeed in approximately $\binom{n}{k}/0.29\binom{n-t}{k}$ iterations, where one iteration of this function has around $O(n^3)$ bit operations. Grover's algorithm uses about square root of the number of iterations, i.e., $\sqrt{\binom{n}{k}/0.29\binom{n-t}{k}}$.

Then the work factor for the Basis Quantum Information Set Decoding attack, which is the complete number of qubit operations for finding a solution, is $O(n^3)\sqrt{\binom{n}{k}/0.29\binom{n-t}{k}}$. Note that the meaning of 0.29 is that, on overage, 29% of the selected matrices G_S are non-singular when G is a generator matrix of the Goppa code.

2.3 Codes Implemented in McEliece type Cryptosystems

After the publication of the original McEliece encryption scheme [31], researchers investigated numerous variants that modify it with different types of codes. In

Table 1. Codes used in McEliece type cryptosystems. Symbols used for current status *: only specific instances are broken; †: NIST submission; ‡: NIST finalist.

N	Code	Proposed by	Current status
1	Binary Goppa codes	McEliece, 1978 [31] Bernstein et al., 2019 [6]	Unbroken as of 2021 Classic McEliece‡
2	GRS codes	Niederreiter, 1986 [35]	Broken in 1992 [46]
3	MRD codes	Gabidulin, 1991 [21] Gabidulin et al., 1995 [20]	Broken in 1995 [23] Broken in 1996 [24]
4	Reed-Muller codes	Sidelnikov, 1994 [45]	Broken in 2007 [33]
5	QC-BCH subcodes	Gaborit, 2005 [22]	Broken in 2010 [38]
6	QC-LDPC codes	Baldi et al., 2007 [3]	Broken in 2008 [16]
7	Wild McEliece	Bernstein et al., 2010 [9]	Broken* in 2014 [14]
8	Wild McEliece Incognito	Bernstein et al., 2011 [10]	Broken* in 2014 [18]
9	Convolutional codes	Löndahl et al., 2012 [30]	Broken in 2013 [27]
10	QC-MDPC codes	Misoczki et al., 2013 [34] Aragon et al., 2019 [1]	Unbroken as of 2021 BIKE†
11	Random linear codes	Wang, 2016 [48]	Broken* in 2019 [15] RLCE† [49]
12	Rank-Metric codes	Aguilar Melchor et al., 2019 [32]	Reduced security 2020 [4] ROLLO†
13	Specific self-dual codes	Domosi et al., 2019 [17]	Not studied

this section, we summarize the main proposals of McEliece type cryptosystems, mentioning the corresponding attacks and security analyses where present.

The summary in Table 1 shows that most of the implementations are broken. The attacks used in the security analysis are mainly *structural attacks*, which succeeded in revealing the private key. The common problems in the broken systems are 1) the use of codes with too much structure and 2) the structure of the public key is not well hidden. The hardness assumption upon which code-based cryptosystems ground their security is the intractability of the problem of *Decoding Random Linear Codes* (DRLC). Research on the computational complexity of this problem dates back to the seminal paper by Berlekamp et al. [5], who proved that DRLC is $\mathcal{NP}-$complete in the worst case. Later works (see, e.g., [11,12,19]) showed that DRLC is closely connected with the problem of *learning parity with noise*. This leads to the widely held belief that DRLC is intractable also in the average case and subsequently to the security assumption underlying code-based cryptosystems. However, when the public key is distinguishable from a random code, such an assumption is no longer true.

From the summarized results, besides the original cryptosystem based on Goppa codes, there is one more unbroken system, BIKE, based on *Quasi-Cyclic*

Moderate Density Parity Check (QC-MDPC) codes. The other two implementations (11 and 12 in Table 1) have some problems. In 11, there are six proposed codes for private keys claimed as random codes, but they have a special structure. In three of these cases, the private key has been retrieved from the public key in polynomial time. Thus, only half of the proposed codes remain for further studies. In 12, the authors of the proposed rank-metric codes have also reported/published an attack that reduces the security level from 256 bits to 200 bits. After this new finding, there is no exact mapping between the parameters of the rank-metric codes and the actual system security level.

Finally, as far as we know, entry 13 is the only published example of a self-dual code implemented in a McEliece type cryptosystem. This code has a very small minimum weight and cannot be considered optimal in this sense. Moreover, there is no extensive cryptanalysis and no defined security of the system. We include it here because we did not find any other examples of a McEliece type cryptosystem based on self-dual codes.

3 McEliece type Cryptosystem using a Binary [104,52,18] Code

We implement an example of a binary $[104, 52, 18]$ self-dual code in a McEliece type cryptosystem. The code is one of the 18 codes given in [26] that has 23 700 codewords of weight 18. The code is denoted by $\mathcal{C}$.

3.1 Cryptosystem

To define the implementation, we follow the description of the McEliece cryptosystem as given in Section 2.

1. System parameters:
   (a) $k = 52$ length of the message m.
   (b) $n = 104$ the length of the ciphertext.
   (c) $t = 8$ the number of the intentionally added errors.
2. *Key generation*: let G be a generator matrix of the $[104, 52, 18]$ self-dual code. Since the public key $G' = SGP$ is expected to be in a systematic form, P is randomly chosen, whereas S is calculated, e.g., from $[GP \mid I_{52}]$ after Gaussian elimination.
   - Choose a random 104×104 permutation matrix P and compute GP. Compute a 52×52 invertible matrix S such that SGP is in a systematic form.
   - Compute $G' = SGP$ and, S^{-1} and P^{-1} - the inverse of P and S.
   - *Public key*: (G', t).
   - *Private key*: (G, P, S).
3. *Encryption*: split the data for encryption into k-bit blocks. Then each block m is encrypted as $r = G'm + e$, where e is a random vector of length n and weight t. Stated differently, the message m is encrypted with the public key (G', t) with t errors intentionally introduced by adding the error vector e.

4. *Decryption:* the decryption steps for the received vector r are:
   (a) Compute $r' = rP^{-1}$, which is $mSG + eP^{-1}$.
   (b) Decode r' into a codeword c' using the decoding Algorithm 2, which is discussed in Section 3.2.
   (c) Compute $c' \in \mathcal{C}$ as $c' = mSG$, and denote by c the first k bits of c' (since G is in systematic form).
   (d) Return $m = cS^{-1}$.

3.2 Decoding Algorithm

The decoding algorithm that we apply in the second step of the decryption phase described in Section 3.1 combines the two algorithms presented in [13] and [29]. From [13], we choose one of the hard-decision deterministic decoding schemes, namely *Algorithm II*, which uses the set of minimum weight codewords of the orthogonal code. This algorithm is generalized in [29] by using any other set of fixed weight dual codewords or a combination of such sets instead of the minimum weight codewords.

First, we define the elements used in the decoding scheme and then the steps of the algorithms. Let $\mathcal{D} \subset \mathbb{F}_2^n$ be an $[n, k, d]$ binary code and $\mathcal{D}^\perp$ be its dual code with minimum distance $d^\perp$. Denote by B the set of all codewords in $\mathcal{D}^\perp$ with weight d_B such that $d_B \geq d^\perp$ (d_B close to $d^\perp$ as in [13]), i.e., $B = \{b \in \mathcal{D}^\perp \mid wt(b) = d_B\}$.

Let $r = c+e$ be the received vector, where $c \in \mathcal{D}$ and $e \in \mathbb{F}_2^n$ is an error vector. Then, for all $b_i \in B$ it follows that $\langle r, b_i\rangle = \langle c+e, b_i\rangle = \langle c, b_i\rangle + \langle e, b_i\rangle = \langle e, b_i\rangle$, due to the fact that c and b_i are orthogonal codewords, hence $\langle c, b_i\rangle = 0$.

Consider $WT_B(r) = \sum_{b_i \in B} \langle r, b_i\rangle$ as the sum of all scalar products $\langle e, b_i\rangle$, with $b_i \in B$. Stated differently, we count how many codewords in B are not orthogonal to the received vector. Algorithm II in [13] is based on the following observation: given two error vectors e_1 and e_2 with weight $wt(e_1) \leq wt(e_2) \leq \frac{d}{2}$, then $WT_B(e_1) \leq WT_B(e_2)$ is valid in most cases (according to [13]). The steps of this decoding scheme are given in Algorithm 2.

In [29], the considered function is a linear combination of functions as $WT_B(r)$. The dual code $\mathcal{D}^\perp$ is split into sets of codewords with the same weight: B_0, B, $\ldots$, B_n for $d_i = 0, 1, \ldots, n$. The counting function equals:

$$U(r) = \sum_{d_i=0}^{n} U_{d_i}(r), \text{ where } U_{d_i}(r) = \alpha_{d_i} WT_{B_i}(r), \tag{6}$$

and where $\alpha_{d_i} \in \mathbb{R}$, called *weighted factor*, can be assumed to be only dependent of the weight d_i of the dual codewords in B_{d_i}. The function $U(r)$ is called *potential function* and U_{d_i} *subpotentials*. According to [29], for efficient decoding it is not necessary to use all subpotentials in the potential function but only some of them. A decoding example presented in [29] is only using the subpotentials of the maximum and minimum weight vectors in $\mathcal{D}^\perp$.

The decoding schemes that we implement are from Algorithm 2, where instead of $WT_B(r)$, we are using $U(r)$ with only one or two subpotentials and with

Algorithm 2: Hard-decision decoding using a set of dual codewords.

1 Denote $v = r$, r-received vector
  Calculate
  $X = WT_B(v)$
2 **if** $X = 0$ **then**
  go to 6)
else
3 Calculate
   $\epsilon_i = WT_B(v + e_i)$ for $i = 1, 2, \ldots, n$,
   where $e_i = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$ with 1 in the i^{-th} coordinate
4 Find $j \in \{1, 2, \ldots, n\}$ with
   $\epsilon_j = min\{\epsilon_i \mid i = 1, 2, \ldots, n\}$
5 $v = v + \epsilon_j$
   $X = \epsilon_j$
   go to 2)
6 Decode r as the codeword v. Exit.

weighted factors always equal to 1. The number of subpotentials and the value for the factors are determined by experimental evaluation.

3.3 Decoding of the $[104, 52, 18]$ Self-dual Code $\mathcal{C}$

Let B_{18} be the set of all codewords in $\mathcal{C}^{\perp} = \mathcal{C}$ of weight 18. The cardinality of B_{18} is 23 700. Moreover, $rank(B_{18}) = 52$, which means that the set B_{18} spans the entire code.

For decoding, we use Algorithm 2 with potential function $U(r) = U_{18}(r) = WT_{B_{18}}(r)$. A programming implementation is tested on 2 000 random examples of received vectors r, where $r = mG + e$ with m a random message of length 52 and e a random error vector of length 104 and $wt(e) = 8$. All vectors r are correctly decoded.

In the setup, the self-dual $[104, 52, 18]$ code $\mathcal{C}$ is a private key of a McEliece type cryptosystem. Then:

1. The rows of a generator matrix G of $\mathcal{C}$ are orthogonal.
2. The matrix GP, P permutation matrix, generates an equivalent to $\mathcal{C}$ self-dual code.
3. The matrix $G' = SGP$, S being the non-singular matrix, consists of rows which are linear combinations of rows in GP, i.e., SGP generates self-dual code with the same minimum weight as in $\mathcal{C}$.

From the last step, it follows that Algorithm 2 can be applied directly on the public key G' and it will decrypt any ciphertext into a message without any additional knowledge. In order to do it, the set of minimum weight codewords generated by G' are required. This set can be obtained for a self-dual code by computing all linear combinations of $1, 2, \ldots, d/2$ rows in G' and in the parity-check matrix of G' when both matrices are in a systematic form.

An attacker to reveal the structure of the public key only needs to check the self orthogonality of G' and when $k = n/2$, then G' generates a self-dual code. Self orthogonality check includes only computing $k(k-1)/2$ inner products. Generating the set of minimum weight codewords in the public key and in the private key takes the same effort, i.e., the attacker has the work equal to the work of the creator of the encryption system.

The number of all linear combinations is:

$$L_{nb} = 2\sum_{i=1}^{d/2}\binom{k}{i} = 2\sum_{i=1}^{9}\binom{52}{i} \approx 2^{33}.$$

We will see later that 33 bits security is much higher than the security level of this system but this approach breaks the system entirely. Therefore, a McEliece type cryptosystem using self-dual codes directly as a private key is vulnerable to a key-recovery attack [2].

To avoid this vulnerability, we consider a $[102, 51, 17]$ punctured code of the code $\mathcal{C}$ for the private key, instead of the complete code $\mathcal{C}$. Let matrix G_{short} be obtained from G by removing two columns and one row. Let also $\mathcal{C}_{short}$ be the punctured code of $\mathcal{C}$ generated by G_{short}. The aim is to preserve the error-correcting capability of 8 errors for $\mathcal{C}_{short}$. To achieve this, the set B_{18} must have in the deleted columns only combinations of $[0, 0]$, $[0, 1]$, or $[1, 0]$. The particular set B_{18}, $B_{18} \subset \mathcal{C}$, has 6 column pairs with this property. G_{short} is obtained from G particularly by removing the first two columns and the first row.

To decode the punctured $[102, 51, 17]$ code $\mathcal{C}_{short}$ with generator matrix G_{short}, we present two strategies: A_1 and A_2. Strategy A_1 is a known procedure to directly decode the punctured code, which is applicable for codes of a small length,. The strategy A_2 is new, applicable only when the number of errors is known, and it decodes the punctured code via the complete code. If there exists a fast decoding scheme for the complete code, the strategy A_2 is suitable for codes of any length.

A_1 Decoding. This strategy performs the decoding via the Algorithm 2, using the potential function $U(r) = U_{17}(r) + U_{18}(r)$. The set B_{18} with the first two columns removed is denoted by B_{18_short}. The elements of B_{18_short}, which are orthogonal to $\mathcal{C}_{short}$ and have weight 17 and weight 18, form the sets B'_{17} and B'_{18}, respectively. These two sets are used to calculate the subpotentials as $U_{17}(r) = WT_{B'_{17}}(r)$ and $U_{18}(r) = WT_{B'_{18}}(r)$.

We obtained $|B'_{17}| = 5\,929$, $|B'_{18}| = 11\,850$, $rank(B'_{17}) = 49$, $rank(B'_{18}) = 50$ and together, $rank(B'_{17} \cup B'_{18}) = 51$. Using $B'_{17} \cup B'_{18}$ in the decoding algorithm, we guarantee that each received vector that is orthogonal to this set will be a codeword of the punctured code.

We tested an implementation of Algorithm 2 with the aforementioned potential function $U(r)$ on a sample of $2\,000$ random received vectors r. In this

[2] The private key structure is revealed, and this fact can be used for direct decoding via the public key.

case $r = mG + e$ with m a random message of length 51 and e a random error vector of length 102 with $wt(e) = 8$. All vectors r are correctly decoded. An experiment shows that using the Algorithm 2 only with B_{17} or only with B_{18} instead of both does not always decode. For B_{17}, there are 238 received vectors out of 2 000 which are not decoded, whereas for B_{18}, there are 9 out of 2 000 also not decoded. It is confirmed that all 347 not decoded vectors are correctly decoded using the Algorithm 2 with $B'_{17} \cup B'_{18}$.

***A_2* Decoding.** Let m be a message of length 51 and $(0 \mid m)$ be m padded with one zero from the left. Denote by P_{short} and S_{short} the permutation and the non-singular matrices used for the public key $G'_{short} = S_{short} G_{short} P_{short}$. The matrix G_{short} is the punctured matrix of G, defined as:

$$G = \begin{pmatrix} g_{1,1} & g_{1,2} & g_{1,3} & \cdots & g_{1,104} \\ g_{2,1} & g_{2,2} & & & \\ g_{3,1} & g_{3,2} & & G_{short} & \\ \vdots & \vdots & & & \\ g_{52,1} & g_{52,2} & & & \end{pmatrix} . \tag{7}$$

One can show that:

$$\begin{aligned} (0 \mid m) \cdot S \cdot G \cdot P = (0 \mid m \cdot S_{short}) \cdot G \cdot P \ &= \\ = (m_1^*, m_2^* \mid mG'_{short}) \ &, \end{aligned} \tag{8}$$

where S includes S_{short}, P includes P_{short} and both are given in Appendix A. Thus, we can decode r' of length 102 via decoding a padded $(*, * \mid r')$ of length 104 by the initial self-dual code $\mathcal{C}$. The decryption including this decoding strategy is described in Algorithm 3. An experiment for Algorithm 3 with 2 000 random examples of received vectors r shows that all of vectors r are correctly decoded and decrypted.

3.4 Cryptanalysis

The attacks described in Section 2.2 are considered against the punctured code of the $[104, 52, 18]$ self-dual code and against the Goppa codes, which would provide a bit security of the McEliece cryptosystem close to the bit security provided by the first code. The chosen Goppa codes are small, only with length $n = 2^m$, $m = 6, 7$ and a number of errors from 4 till 10. The choice of parameters for Goppa codes is also restricted by the information rate $R > 0.4$, $R = k/n$, since the code has to be efficient, i.e., $n - k$ check bits do not exceed much the k information bits.

Algorithm 3: Decryption using padded ciphertext.

1 Denote
   $s = [[0,0],[0,1],[1,0],[1,1]]$,
   $i = 1$, $t = 8$, $k = 51$, $n = 102$
2 Compute
   $r' = rP^{-1}$, r-received vector of length k
3 **while** $i < 5$
4   Pad
     r' into $(s[i] \mid r')$
5   Decode
     $(s[i] \mid r')$ into c_1, $c_1 \in \mathcal{C}$, by Algorithm 2 with decoding set B_{18}.
6   **if** 5) successful **then**
7   Denote
       $c_2 = c_1[3 : n+2]$, $m_2 = c_2[1 : k]$
8   Compute
       $m_1 = m_2 * S^{-1}$
9   **if** $(m_1 \in \mathcal{C}'_{short} \quad \wedge \quad \text{weight}(m_1 * G'_{short} - r') == t)$ **then**
       Decrypt r as m_1. **Exit**.
10   $i = i + 1$ increase the index
11 **if** $i == 5$ **then**
   return 'Unsuccessful decryption'. **Exit**.

Name	Attack
A_1	Brute force attack towards the message
A_2	Brute force attack towards the coset leaders of the private key
A_3	Brute force attack on the error-vector
A_4	Basis Information Set Decoding attack
A_5	Stern's attack
A_6	Basis Quantum Information Set Decoding attack

The total cost for each attack is defined in Section 2.2. In Table 2, we list the values of log_2 of the total cost for each of the attacks. The notations in Table 2 are defined in the list above. For the attacks A_4 and A_6, the value of the parameter β equals $29,05\%$.

As discussed in the previous section, using a self-dual code for a private key in a McEliece type cryptosystem is not secure. Instead, a punctured code is considered. The values in Table 2 show that the classical bit security of the $[102, 51, 17]$ code C_{ii} is 22.25 bits and the Goppa codes with the closest security level are C_3 and C_4. For the quantum security level our code example is closest to the code C_8. Comparing the size of C_{ii} with the sizes of all three Goppa codes, C_3, C_4, and C_8, one can show that the size of C_{ii} is at least 28% smaller than the sizes of C_3, C_4, and C_8.

Remark 1. Structural attacks are not considered because both the public and the private keys do not have any specific structure. In order to reconstruct the private key to the initial self-dual code, $2k + (n + 2)$ bits have to be restored, which has a much higher work factor than the claimed security level requires.

Table 2. $log_2(Work\ factor)$ of different attacks.

Goppa codes											
code	n	k	t	k(n-k)	A_1	A_2	A_3	A_4	A_5	A_6	$min_(A_1, ..., A_5)$
C_1	128	100	4	2 800	100	28	23.3468	30.7427	20.2171	25.3371	20.5533
C_2	128	93	5	3 255	93	35	27.9791	31.074	21.1199	25.3457	21.1199
C_3	128	86	6	**3 612**	86	42	32.3366	31.0785	21.9873	25.1787	**21.9873**
C_4	128	79	7	**3 871**	79	49	36.46	30.801	22.6618	24.8562	**22.6618**
C_5	128	72	8	4 032	72	56	40.3789	30.2708	23.19	24.3903	23.3368
C_6	128	65	9	4 095	65	63	44.1158	29.5066	23.5314	23.7869	23.5629
C_7	128	58	10	4 060	58	70	47.6887	28.5193	23.7787	23.0466	23.7787
C_8	128	51	11	3 927	51	77	51.1119	27.3117	23.8866	22.1645	23.8866
C_9	64	52	2	624	52	12	10.9773	23.8201	14.7128	20.4607	10.9773
C_{10}	64	46	3	828	46	18	15.3465	24.0306	16.7361	20.3007	15.3465
C_{11}	64	40	4	960	40	24	19.2773	23.6536	17.9063	19.8097	17.9063
C_{12}	64	34	5	1 020	34	30	22.8622	22.7898	18.5835	19.0261	18.5835
C_{13}	64	28	6	1 008	28	36	26.1599	21.4744	18.9469	17.9482	18.9469
The self-dual code with a punctured code derived from it											
C_i	104	52	8	2 704	52	52	37.9062	27.3062	22.3401	22.2038	22.3401
C_{ii}	102	51	8	**2 601**	51	51	37.6741	27.2311	22.253	22.1242	**22.253**

4 Parameters Estimation for Self-dual Codes with Bit Security 80, 128, and 256

To estimate parameters for the self-dual codes, which would provide a security level of 80, 128, and 256 bits, we apply the upper bounds for the work factor of the attacks in the previous section to the known recently proposed Goppa codes with these security levels. Since our attacks are not the best known, we expect to obtain higher values for the upper bounds. These higher values we use further for the estimation of the parameters of the self-dual codes.

The private key of the original McEliece cryptosystem is a $[1\,024, 525]$ Goppa code with the error-correcting capability of 50 errors. It is initially estimated to provide security of 64 bits. Latter, via an improved version of Stern's attack in [8] the security of the system is reduced to 60.5 bits. In the same publication, the authors proposed parameters for the Goppa codes, where implementation in the McEliece cryptosystem would provide a security level of 80, 128, and 256 bits. The proposed codes are listed in Table 3. The latest proposed codes providing security levels of 128, 196, and 256 bits are in the NIST proposal [6].

From the results listed in Table 3, it follows that we have to search for codes providing a bit security level of 83, 148, and 302 to ensure that they would provide at least 80, 128, and 256 bits security concerning the latest attacks. In Table 4, we list the parameters of a few such codes. A larger list is included in Table 5 in Appendix B.

Note that these are the parameters of the punctured $[n, k, 2t+1]$ codes. The corresponding self-dual codes have to be with length $n+2$ and minimum weight $2t+3$ to ensure that the punctured codes are within the required parameters.

Table 3. $\min(Log_2(Workfactor))$ of the attacks $A_1, \ldots, A_6$ in Section 3.4.

Goppa codes							
code	security	n	k	t	k(n-k)	$min_(A_1, ..., A_5)$	A_6
D_1	80 [8]	1 632	1 269	34	460 647	82.231	69.5887
D_2	128 [8]	2 960	2 288	57	1 537 536	129.8371	96.7078
D_3	128 [6]	3 488	2 720	64	2 088 960	147.4275	106.5127
D_4	256 [8]	6 624	5 129	117	7 667 855	259.2255	166.1179
D_5	256 [6]	6 688	5 024	128	8 359 936	265.2662	168.9545
D_6	256 [6]	6 960	5 413	119	8 373 911	266.0612	169.8205
D_7	256 [6]	8 192	6 528	128	10 862 592	302.1663	188.9797

The estimation for the self-dual codes is for the minimum weight with 15% less than the upper bounds for the minimum weight of a putative self-dual code: $d_1 \leq 4\lfloor \frac{n_1}{24} \rfloor + 4, \quad if \quad n_1 \not\equiv 22 \pmod{24}$, and $d_1 \leq 4\lfloor \frac{n_1}{24} \rfloor + 6, \quad if \quad n_1 \equiv 22 \pmod{24}$ for a self-dual $[n_1, n_1/2, d_1]$ code [42].

This restriction increases the probability that such a code if it exists, is not unique and could be constructed. The existence of a large number of codes of the same family is a preliminary requirement for the security of the McEliece type cryptosystem.

The size of the putative punctured codes B_1, B_9, and B_{31} is at least 38% smaller than the size of the proposed smallest Goppa codes D_1, D_2, and D_4 providing the security level of 80, 128, and 256 bits, correspondingly. In the next section, we will present a possible construction of a self-dual code where the punctured code has the parameters of B_1.

Table 4. $\min(Log_2(Workfactor))$ of the attacks $A_1, \ldots, A_6$ in Section 3.4.

Punctured codes						
code	n	k	t	k(n-k)	$min_(A_1, ..., A_5)$	A_6
B_1	1 062	531	75	281 961	87.3248	67.5796
B_2	1 064	532	75	283 024	87.3264	67.5837
B_8	1 076	538	75	289 444	87.2886	67.6079
B_9	1 894	947	134	896 809	147.8721	101.2093
B_{10}	1 896	948	134	898 704	147.869	101.2097
B_{30}	1 940	970	136	940 900	149.8767	102.3316
B_{31}	4 006	2 003	284	4 012 009	303.9682	183.5916
B_{32}	4 008	2 004	284	4 016 016	303.9619	183.5895
B_{42}	4 028	2 014	284	4 056 196	303.8758	183.5694

5 A New Example of McEliece type Cryptosystem with 80-bit Security

To construct a McEliece type cryptosystem, we first define an example of a binary $[1\,064, 532, d \geq 168]$ self-dual code, then a punctured code of it as a private key for the scheme. At last, an efficient decoding scheme as a part of the decryption process is discussed.

5.1 A Binary $[1\,064, 532, d \geq 162]$ Self-dual Code

For constructing a binary $[1\,064, 532, d \geq 162]$ self-dual code we use a known algorithm presented in [43] and [44]. Details about it are included in Appendix C. Here, we provide only a summary.

Let B be a self-dual $[1\,064, 532, d \geq 162]$ code having an automorphism σ of order 133 with 8 cycles of length 133 and no fixed points. Without loss of generality σ can be represented as: $\sigma = \Omega_1\Omega_2\ldots\Omega_8$, where Ω_i is a cycle of length 133 for $1 \leq i \leq 8$.

Then, for the code B the following holds [43]:

1. $B = F_\sigma(B) \oplus E_\sigma(B)$,
2. the fixed subcode $\pi(F_\sigma(B))$ is a binary $[8, 4]$ self-dual code, and
3. the vectors of image $\varphi(E_\sigma(B))$ are from $\mathcal{P}^8$, where $\mathcal{P}$ is the set of even weight polynomials in $\mathbb{F}_2/(x^{133} - 1)$.

The sets $F_\sigma(B)$, $E_\sigma(B)$, and the images π and φ are defined in Appendix C. First, generator matrices X and Y of $F_\sigma(B)$ and $E_\sigma(B)$ are constructed and then a generator matrix of the code B as

$$G = \begin{pmatrix} X \\ Y \end{pmatrix} \quad . \tag{9}$$

Both matrices X and Y in Eq. 9 are included in Appendix C.

Due to computation time, the minimum weight of the code is not confirmed to be greater or equal to 162. All linear combinations of up to 8 vectors of G and the corresponding parity-check matrix are computed. They all have a weight greater than or equal to 168. A random linear combination of a random number of rows of G on a single 16 RAM Intel7 PC for 30 days did not result in a vector with a smaller weight than 168.

5.2 McEliece Type Cryptosystem Using the New Code Example

Let B_1 be a punctured $[1\,062, 531, d' \geq 160]$ code obtained from the self-dual code B_1 by removing the first two columns and the first row. Let us denote a generator matrix of B_1 by M. This matrix will be used for a private key of the system.

1. System parameters:
   (a) $k = 531$ length of the message m.
   (b) $n = 1\,062$ the length of the ciphertext.

(c) $t = 80$ the number of the intentionally added errors.

2. *Key generation:* M - a generating matrix of code B_1; P a random $1\,031 \times 1\,031$ permutation matrix; S - a non singular dense 531×531 matrix such that $G' = SMP$ is in a systematic form. Compute $G' = SMP$ and, S^{-1} and P^{-1} - the inverse of P and S.
Public key (G', t)
Private key (M, P, S).
3. *Encryption:* $r = G'm + e$ where m is a message block of length 531 and e is the intentionally added random error vector of length 1 062 and weight 80.
4. *Decryption:* the decoding Algorithm 2 applied for the small example code of length 102 is using the set of the minimum weight dual codewords or union of sets with chosen weights from the dual code. For the code B_1 to find all the codewords with the minimum weight is a computationally difficult problem. Additionally, the set can be very large, i.e., it requires a large memory, which is a limitation for practical implementation in the current communication systems. A decoding algorithm for self-dual codes with the same construction as the code B is recently introduced in [50]. It uses a smaller set of codewords with a weight equal to or slightly higher than the minimum weight. This decoding scheme is used in Algorithm 3.

Remark 2. Due to time limitations, we could not complete the simulations to determine an optimal decoding set of codewords.

An example of a self-dual $[266, 133, 36]$ code, constructed via an automorphism of order 133 as the code B, is included in [50]. Using a set of only 2 614 codewords the mentioned decoding algorithm corrects up to $t-2$ errors in 100% of the cases, where $t = 17$.

Note that the minimum weight of the punctured code B_1 is 160, which means B_1 has an error-correcting capability of up to 79 errors. According to the estimation in Section 4 for security level of 80 bits, the code B_1 needs to correct 75 errors, which is $t - 4$. As such, we expect that the algorithm will provide decoding with the same or close to this efficiency when using a large enough decoding set of codewords.

6 Conclusions

This paper proposes a McEliece type cryptosystem using high minimum distance self-dual codes and punctured codes derived from them. First, we provide a small example of the cryptosystem using a code obtained from an optimal self-dual code of length 104. Next, we determine the parameters of a putative optimal self-dual code, which, if implemented in a McEliece type cryptosystem, would provide a classic security level of 80, 128, and 256 (quantum 67, 101, and 183) bits, respectively. For the 80-bit security case, we construct an optimal self-dual code of length 1 064, achieving a reduction of the key size of around 38.5% compared to the original McEliece cryptosystem. Since we proposed a new McEliece type cryptosystem, there are several directions to follow in future work. We believe the next step should include further investigation concerning efficient software implementation and run-time analysis.

References

1. Aragon, N., Barreto, P., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Gueron, S., Guneysu, T., Melchor, C.A., et al.: Bike: bit flipping key encapsulation (2017), http://bikesuite.org, accessed 15 June 2021
2. Baldi, M.: QC-LDPC Code-Based Cryptography (5.4 Cryptanalysis of the McEliece and Niederreiter Cryptosystems). Springer (2014)
3. Baldi, M., Chiaraluce, F.: Cryptanalysis of a new instance of mceliece cryptosystem based on qc-ldpc codes. In: IEEE International Symposium on Information Theory (ISIT 2007). p. 2591 – 2595 (2007)
4. Bardet, M., Briaud, P., Bros, M., Gaborit, P., Neiger, V., Ruatta, O., Tillich, J.: An algebraic attack on rank metric code-based cryptosystems. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12107, pp. 64–93. Springer (2020)
5. Berlekamp, E.R., McEliece, R.J., van Tilborg, H.C.: On the inherent intractability of certain coding problems http://authors.library.caltech.edu/5607/1/BERieeetit78.pdf. IEEE Transactions on Information Theory **24**(3), 384–386 (1978)
6. Bernstein, D., Chou, C., Lange, T., von Maurich, I., Misoczki, R., Niederhagen, R., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., Szefer, J., Wang, W.: Classic mceliece: conservative code-based cryptography. (2017), https://classic.mceliece.org/nist/mceliece-20201010.pdf
7. Bernstein, D.J.: Grover vs. mceliece. Proceedings of the Third International Conference on Post-Quantum Cryptography p. 73–80 (2010)
8. Bernstein, D.J., Lange, T., Peters, C.: Attacking and defending the mceliece cryptosystem. In: Buchmann, J., Ding, J. (eds.) Post-Quantum Cryptography. pp. 31–46. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
9. Bernstein, D.J., Lange, T., Peters, C.: Wild mceliece. In: Selected Areas in Cryptography. pp. 143–158. Springer Berlin Heidelberg (2011)
10. Bernstein, D.J., Lange, T., Peters, C.: Wild mceliece incognito. In: Post-Quantum Cryptography. pp. 244–254. Springer Berlin Heidelberg (2011)
11. Blum, A., Furst, M.L., Kearns, M.J., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: Stinson, D.R. (ed.) Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings. Lecture Notes in Computer Science, vol. 773, pp. 278–291. Springer (1993)
12. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM **50**(4), 506–519 (2003)
13. Bossert, M., Hergert, F.: Hard- and soft-decision decoding beyond the half minimum distance—an algorithm for linear codes (corresp.). IEEE Transactions on Information Theory **32**(5), 709–714 (1986)
14. Couvreur, A., Otmani, A., Tillich, J.P.: Polynomial time attack on wild mceliece over quadratic extensions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 17–39. Springer (2014)

15. Couvreur, A., Lequesne, M., Tillich, J.: Recovering short secret keys of RLCE in polynomial time. In: Ding, J., Steinwandt, R. (eds.) Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers. Lecture Notes in Computer Science, vol. 11505, pp. 133–152. Springer (2019)
16. Couvreur, A., Otmani, A., Tillich, J.P.: Polynomial time attack on wild mceliece over quadratic extensions. In: Advances in Cryptology – EUROCRYPT 2014. pp. 17–39. Springer Berlin Heidelberg (2014)
17. Domosi, P., Hannusch, C., Horváth, G.: A cryptographic system based on a new class of binary error-correcting codes. Tatra Mountains Mathematical Publications **73**, 83–96 (2019)
18. Faugère, J.C., Perret, L., de Portzamparc, F.: Algebraic attack against variants of mceliece with goppa polynomial of a special form. In: Advances in Cryptology – ASIACRYPT 2014. pp. 21–41. Springer Berlin Heidelberg (2014)
19. Feldman, V., Gopalan, P., Khot, S., Ponnuswami, A.K.: New results for learning noisy parities and halfspaces. In: 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings. pp. 563–574. IEEE Computer Society (2006)
20. Gabidulin, E.M.: On public-key cryptosystems based on linear codes. In: Proc. of 4th IMA Conference on Cryptography and Coding 1993. pp. 482–489. IMA Press, Southend-on Sea (1995)
21. Gabidulin, E.M., Paramonov, A.V., Tretjakov, O.V.: Ideals over a non-commutative ring and their application in cryptology. In: Advances in Cryptology — EUROCRYPT '91. pp. 482–489. Springer Berlin Heidelberg (1991)
22. Gaborit, P.: Shorter keys for code based cryptography. In: International Workshop on Coding and Cryptography (WCC 2005). p. 81 – 91 (2005)
23. Gibson, J.K.: Severely denting the gabidulin version of the mceliece public key cryptosystem. Designs, Codes and Cryptography **6**(1), 37 – 45 (1995)
24. Gibson, J.K.: The security of the gabidulin public key cryptosystem. In: Advances in Cryptology — EUROCRYPT '96. pp. 212–223. Springer Berlin Heidelberg (1996)
25. Hamdaoui, Y., Sendrier, N.: A non asymptotic analysis of information set decoding. Cryptology ePrint Archive, Report 2013/162 (2013), https://eprint.iacr.org/2013/162
26. Harada, M., Kiermaier, M., Wassermann, A., Yorgova, R.: New binary singly even self-dual codes. IEEE Transactions on Information Theory **56**(4), 1612–1617 (2010)
27. Landais, G., Tillich, J.: An efficient attack of a mceliece cryptosystem variant based on convolutional codes. CoRR **abs/1302.5120** (2013), http://arxiv.org/abs/1302.5120
28. Lee, P.J., Brickell, E.F.: An observation on the security of mceliece's public-key cryptosystem. In: Barstow, D., Brauer, W., Brinch Hansen, P., Gries, D., Luckham, D., Moler, C., Pnueli, A., Seegmüller, G., Stoer, J., Wirth, N., Günther, C.G. (eds.) Advances in Cryptology — EUROCRYPT '88. pp. 275–280. Springer Berlin Heidelberg, Berlin, Heidelberg (1988)
29. Lohnert, R.: Potential-decoding, error correction beyond the half minimum distance for linear block codes. In: Proceedings of 1995 IEEE International Symposium on Information Theory. pp. 52– (1995)
30. Löndahl, C., Johansons, T.: A new version of mceliece pkc based on convolutional code. Information and Communications Security (ICICS) **LNCS 7168**, 461–470 (2012)

31. McEliece, R.: A public-key cryptosystem based on algebraic coding theory. Deep Space Network Progress Report **44**, 114–116 (1978), https://ipnpr.jpl.nasa.gov/progress_report/42-44/44N.PDF
32. Melchor, C.A., Aragon, N., Bardet, M., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Hauteville, A., Otmani, A., Ruatta, O., Tillich, J.P., Zémor, G.: ROLLO-Rank-Ouroboros, LAKE & LOCKER (2019), https://pqc-rollo.org, accessed 15 June 2021
33. Minder, L., Shokrollahi, A.: Cryptanalysis of the sidelnikov cryptosystem. In: Advances in Cryptology - EUROCRYPT 2007. pp. 347–360. Springer Berlin Heidelberg (2007)
34. Misoczki, R., Tillich, J.P., Sendrier, N., Barreto, P.S.: Mdpc-mceliece: New mceliece variants from moderate density parity-check codes. In: IEEE International Symposium on Information Theory (ISIT 2013). pp. 2069–2073 (2013)
35. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Probl. Control and Inform. Theory pp. 159–166 (1986)
36. NIST: Recommendation for key management. NIST (2012), https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-57p1r3.pdf
37. NIST: Status report on the second round of the nist post-quantum cryptography standardization process. NIST (July 2020), https://csrc.nist.gov/publications/detail/nistir/8309/final
38. Otmani, A., Tillich, J.P., Dallot, L.: Cryptanalysis of two mceliece cryptosystems based on quasi-cyclic codes. Mathematics in Computer Science **3**(2), 129 – 140 (2010)
39. Peters, C.: Information-set decoding for linear codes over fq. In: Sendrier, N. (ed.) Post-Quantum Cryptography. pp. 81–94. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
40. Pless, V.: Decoding the golay codes. IEEE Transactions on Information Theory **32**(4), 561–567 (1986)
41. Prange, E.: The use of information sets in decoding cyclic codes. IRE Transactions on Information Theory, **vol. 8, no. 5, pp. 5-9** (1962)
42. Rains, E.M.: Shadow bounds for self-dual codes. IEEE Trans. Inform. Theory **44**, 134–139 (1998)
43. R.Dontcheva, Zanten, A., S.M.Dodunekov: Binary self-dual codes with automorphisms of composite order. IEEE Trans.of Inf.Theory **50**(2), 311–318 (2004)
44. R.Yorgova: On binary self-dual codes with automorphisms. IEEE Trans.of Inf.Theory **54**(7), 3345–3351 (2008)
45. Sidelnikov, V.M.: A public-key cryptosystem based on binary reed-muller codes. Discrete Mathematics and Applications **4**(3), 191 – 208 (1994)
46. Sidelnikov, V.M., Shestakov, S.O.: On insecurity of cryptosystems based on generalized reed-solomon codes. Discrete Mathematics and Applications **2**(4), 439–444 (1992)
47. Stern, J.: A method for finding codewords of small weight. In: Cohen, G.D., Wolfmann, J. (eds.) Coding Theory and Applications, 3rd International Colloquium, Toulon, France, November 2-4, 1988, Proceedings. Lecture Notes in Computer Science, vol. 388, pp. 106–113. Springer (1988)
48. Wang, Y.: Quantum resistant random linear code based public key encryption scheme rlce. In: 2016 IEEE International Symposium on Information Theory (ISIT). pp. 2519–2523 (2016)
49. Wang, Y.: Quantum resistant public key encryption scheme rlce and ind-cca2 security for mceliece schemes. Cryptology ePrint Archive, Report 2017/206 (2017), https://eprint.iacr.org/2017/206

50. Yorgova, R.: On decoding of a specific type of self-dual codes. Computer Science ePrint Archive (2021), https://arxiv.org/

A Defining P and S for Strategy A_2

The matrices S and P referred in Section 3.3 for the A_2 decoding strategy are defined as follows:

$$S = \begin{pmatrix} 1\,0 & \dots & 0 \\ 0 & & \\ \vdots & S_{short} & \\ & & \\ 0 & & \end{pmatrix} \; ; \; P = \begin{pmatrix} 1\,0\,0 & \dots & 0 \\ 0\,1\,0 & \dots & 0 \\ 0\,0 & & \\ \vdots\;\vdots & P_{short} & \\ & & \\ 0\,0 & & \end{pmatrix} . \tag{10}$$

B Parameters of Punctured Codes Derived from Self-dual Codes for Bit Security 80, 128, and 256

In this section, we give work factors for the attacks $A_1, \dots, A_5$. The results are given in Table 5.

C Generating a Binary $[1\,064, 532, d \geq 162]$ Self-dual Code

As already mentioned, for constructing a binary $[1\,064, 532, d \geq 162]$ self-dual code, we use a method presented in [43] and [44]. Let B be a self-dual $[1\,064, 532, d \geq 162]$ code having an automorphism σ of order 133 with 8 cycles of length 133 and no fixed points, i.e., σ has the form: $\sigma = \Omega_1 \Omega_2 \dots \Omega_8$, where Ω_i is a cycle of length 133 for $1 \leq i \leq 8$.

If $v \in B$, then v can be presented as $v = (v|\Omega_1, v|\Omega_2, \dots, v|\Omega_8)$, where $v|\Omega_i = (v_0, v_1, \dots, v_{132})$ denotes the coordinates of v in the i^{-th} cycle of σ. Let further $F_\sigma(B)$ and $E_\sigma(B)$ be defined as $F_\sigma(B) = \{v \in B| \;\; v\sigma = v\}$ and $E_\sigma(B) = \{v \in B| \;\; \mathrm{wt}(v|\Omega_i) \equiv 0 \,(\mathrm{mod}\, 2), \;\; i = 1, \dots, 8\}$.
It is known that both, $F_\sigma(B)$ and $E_\sigma(B)$, are linear subcodes of B. Moreover, $B = F_\sigma(B) \oplus E_\sigma(B)$, where $\oplus$ stands for the direct sum of linear subspaces [43]. Then a generator matrix of B can be decomposed as:

$$G = \begin{pmatrix} X \\ Y \end{pmatrix}, \tag{11}$$

where X is a generator matrices of $F_\sigma(B)$ and Y is a generator matrix of $E_\sigma(B)$.

The map π is defined as:

$$\pi\colon F_\sigma(B) \to \mathbb{F}_2^8, \qquad \pi(v|\Omega_i) = v_j$$

Table 5. $\min(Log_2(Workfactor))$ of the attacks $A_1, \ldots, A_5$ in Section 3.4. $M = \{A_1, ..., A_5\}$. The horizontal lines delimit 80, 128, and 256 bit security levels.

Punctured codes											
code	n	k	t	k(n-k)	$min(M)$	code	n	k	t	k(n-k)	$min(M)$
B_1	1 062	531	75	281 961	87.3248	B_{22}	1 924	962	136	925 444	149.9394
B_2	1 064	532	75	283 024	87.3264	B_{23}	1 926	963	136	927 369	149.9266
B_3	1 066	533	75	284 089	87.3118	B_{24}	1 928	964	136	929 296	149.9236
B_4	1 068	534	75	285 156	87.3136	B_{25}	1 930	965	136	931 225	149.9108
B_5	1 070	535	75	286 225	87.299	B_{26}	1 932	966	136	933 156	149.9078
B_6	1 072	536	75	287 296	87.3009	B_{27}	1 934	967	136	935 089	149.8952
B_7	1 074	537	75	288 369	87.2865	B_{28}	1 936	968	136	937 024	149.8922
B_8	1 076	538	75	289 444	87.2886	B_{29}	1 938	969	136	938 961	149.8796
B_9	1 894	947	134	896 809	147.8721	B_{30}	1 940	970	136	940 900	149.8767
B_{10}	1 896	948	134	898 704	147.869	B_{31}	4 006	2 003	284	4 012 009	303.9682
B_{11}	1 898	949	134	900 601	147.8561	B_{32}	4 008	2 004	284	4 016 016	303.9619
B_{12}	1 900	950	134	902 500	147.853	B_{33}	4 010	2 005	284	4 020 025	303.9509
B_{13}	1 902	951	134	904 401	147.8402	B_{34}	4 012	2 006	284	4 024 036	303.9446
B_{14}	1 904	952	134	906 304	147.8371	B_{35}	4 014	2 007	284	4 028 049	303.9336
B_{15}	1 906	953	134	908 209	147.8244	B_{36}	4 016	2 008	284	4 032 064	303.9273
B_{16}	1 908	954	134	910 116	147.8214	B_{37}	4 018	2 009	284	4 036 081	303.9163
B_{17}	1 910	955	134	912 025	147.8088	B_{38}	4 020	2 010	284	4 040 100	303.9101
B_{18}	1 912	956	134	913 936	147.8058	B_{39}	4 022	2 011	284	4 044 121	303.8991
B_{19}	1 918	959	136	919 681	149.9586	B_{40}	4 024	2 012	284	4 048 144	303.8929
B_{20}	1 920	960	136	921 600	149.9554	B_{41}	4 026	2 013	284	4 052 169	303.8819
B_{21}	1 922	961	136	923 521	149.9425	B_{42}	4 028	2 014	284	4 056 196	303.8758

for some $j \in \Omega_i, i = 1, 2, \ldots, 8$. According to [43], $\pi(F_\sigma(B))$ is a binary self-dual code of length 8. Therefore, a possible generator matrix of $F_\sigma(B)$ is the matrix:

$$X = \begin{pmatrix} s\ o\ o\ o\ o\ s\ s\ s \\ o\ s\ o\ o\ s\ o\ s\ s \\ o\ o\ s\ o\ s\ s\ o\ s \\ o\ o\ o\ s\ s\ s\ s\ o \end{pmatrix},$$

where $s = (1, 1, \ldots, 1)$ is the all ones vector and o is the zero vector in $\mathbb{F}_2^{133}$.

Let $\mathcal{P}$ denote the set of even-weight polynomials in $\mathcal{R} = \mathbb{F}_2[x]/(x^{133} - 1)$ and map φ be the following:

$$\varphi : \ E_\sigma(B) \quad \to \quad \mathcal{P}^8, \tag{12}$$

where $v|\Omega_i = (v_0, v_1, \ldots, v_{132})$ is identified with the polynomial $\varphi(v|\Omega_i)(x) = v_0 + v_1 x + \cdots + v_{132} x^{132}$ in $\mathcal{P}$ for $1 \leq i \leq 8$.

An inner product in $\mathcal{P}^8$ is defined as:

$$\langle g, h \rangle = g_1(x) h_1(x^{-1}) + \cdots + g_8(x) h_8(x^{-1}) \tag{13}$$

for all $g, h \in \mathcal{P}^8$. The image $\varphi(E_\sigma(\mathcal{C}))$ is a self-orthogonal code [44], i.e.,

$$u_1(x) v_1(x^{-1}) + \cdots + u_8(x) v_8(x^{-1}) = 0, \tag{14}$$

for all $u, v \in \varphi(E_\sigma(B))$.

This orthogonality and the factorization of $x^{133}-1$ is used in constructing a generator matrix Y' of $\varphi(E_\sigma(B))$. A possible variant is the following one:

$$Y' = \begin{pmatrix}
e_1(x) & 0 & 0 & 0 & 0 & \alpha_1(x) & \alpha_1(x) & \alpha_1(x) \\
0 & e_1(x) & 0 & 0 & \alpha_1(x) & 0 & \alpha_1^2(x) & \alpha_1^3(x) \\
0 & 0 & e_1(x) & 0 & \alpha_1(x) & \alpha_1^2(x) & 0 & \alpha_1^2(x) \\
0 & 0 & 0 & e_1(x) & 0 & \alpha_1^2(x) & \alpha_1^3(x) & \alpha_1^4(x) \\
0 & \alpha_2(x) & \alpha_2(x) & 0 & e_2(x) & 0 & 0 & 0 \\
\alpha_2(x) & 0 & \alpha_2^2(x) & \alpha_2^2(x) & 0 & e_2(x) & 0 & 0 \\
\alpha_2(x) & \alpha_2^2(x) & 0 & \alpha_2^3(x) & 0 & 0 & e_2(x) & 0 \\
\alpha_2(x) & \alpha_2^3(x) & \alpha_2^2(x) & \alpha_2^4(x) & 0 & 0 & 0 & e_2(x) \\
e_3(x) & 0 & 0 & 0 & 0 & \alpha_3(x) & \alpha_3^2(x) & \alpha_3^3(x) \\
0 & e_3(x) & 0 & 0 & \alpha_3^2(x) & 0 & \alpha_3^3(x) & \alpha_3^5(x) \\
0 & 0 & e_3(x) & 0 & \alpha_3^7(x) & \alpha_3^{13}(x) & 0 & \alpha_3^{17}(x) \\
0 & 0 & 0 & e_3(x) & \alpha_3^5(x) & \alpha_3^{21}(x) & \alpha_3^{23}(x) & 0 \\
0 & \alpha_4^2(x) & \alpha_4^7(x) & \alpha_4^5(x) & e_4(x) & 0 & 0 & 0 \\
\alpha_4(x) & 0 & \alpha_4^{13}(x) & \alpha_4^{21}(x) & 0 & e_4(x) & 0 & 0 \\
\alpha_4^2(x) & \alpha_4^3(x) & 0 & \alpha_4^{23}(x) & 0 & 0 & e_4(x) & 0 \\
\alpha_4^3(x) & \alpha_4^5(x) & \alpha_4^{17}(x) & 0 & 0 & 0 & 0 & e_4(x) \\
e_5(x) & 0 & 0 & 0 & 0 & \alpha_5(x) & \alpha_5^2(x) & \alpha_5^3(x) \\
0 & e_5(x) & 0 & 0 & \alpha_5^2(x) & 0 & \alpha_5^3(x) & \alpha_5^7(x) \\
0 & 0 & e_5(x) & 0 & \alpha_5^5(x) & \alpha_5^{11}(x) & \alpha_5^{13}(x) & \alpha_5^{17}(x) \\
0 & 0 & 0 & e_5(x) & \alpha_5^7(x) & \alpha_5^{21}(x) & \alpha_5^{23}(x) & 0 \\
0 & \alpha_6^2(x) & \alpha_6^5(x) & \alpha_6^7(x) & e_6(x) & 0 & 0 & 0 \\
\alpha_6(x) & 0 & \alpha_6^{11}(x) & \alpha_6^{21}(x) & 0 & e_6(x) & 0 & 0 \\
\alpha_6^2(x) & \alpha_6^3(x) & \alpha_6^{13}(x) & \alpha_6^{23}(x) & 0 & 0 & e_6(x) & 0 \\
\alpha_6^3(x) & \alpha_6^7(x) & \alpha_6^{17}(x) & 0 & 0 & 0 & 0 & e_6(x) \\
e_7(x) & 0 & 0 & 0 & 0 & \alpha_7(x) & \alpha_7^2(x) & \alpha_7^7(x) \\
0 & e_7(x) & 0 & 0 & \alpha_7^{13}(x) & 0 & \alpha_7^{27}(x) & \alpha_7^{31}(x) \\
0 & 0 & e_7(x) & 0 & \alpha_7^3(x) & \alpha_7^5(x) & 0 & \alpha_7^{11}(x) \\
0 & 0 & 0 & e_7(x) & \alpha_7^{17}(x) & \alpha_7^7(x) & \alpha_7(x) & 0 \\
0 & \alpha_8^{13}(x) & \alpha_8^3(x) & \alpha_8^{17}(x) & e_8(x) & 0 & 0 & 0 \\
\alpha_8(x) & 0 & \alpha_8^5(x) & \alpha_8^7(x) & 0 & e_8(x) & 0 & 0 \\
\alpha_8^2(x) & \alpha_8^{27}(x) & 0 & \alpha_8(x) & 0 & 0 & e_8(x) & 0 \\
\alpha_8^7(x) & \alpha_8^{31}(x) & \alpha_8^{11}(x) & 0 & 0 & 0 & 0 & e_8(x) \\
e_9(x) & 0 & 0 & 0 & \alpha_9(x) & \alpha_9(x) & \alpha_9^{319}(x) & \alpha_9^{233370}(x) \\
0 & e_9(x) & 0 & 0 & \alpha_9^2(x) & \alpha_9^2(x) & \alpha_9(x) & \alpha_9^{49}(x) \\
\alpha_9^{512}(x) & \alpha_9^{1024}(x) & \alpha_9^{1139}(x) & 0 & e_9(x) & 0 & 0 & 0 \\
\alpha_9^{512}(x) & \alpha_9^{1024}(x) & \alpha_9^{149579}(x) & \alpha_9^{338}(x) & 0 & e_9(x) & 0 & 0
\end{pmatrix}$$

where the coefficients of the polynomials $e_i(x)$ and $\alpha_i(x)$ for $i = 1, 2, \ldots, 9$ are given in Table 6. Each of the entry polynomials in Y' generates a right circulant 3×133 matrix for the first 8 rows in Y' and a 18×133 right circulant matrix for the rest of 28 rows in Y'. The corresponding matrix with the circulants is the generator matrix Y of $E_\sigma(B)$, i.e., $Y = \begin{pmatrix} y_{1,1} & y_{1,8} \\ \vdots & \vdots \\ y_{36,1} & y_{36,8} \end{pmatrix}$, where $y_{i,j}$ are right-circulant 3×133 cells for the first 8 rows in Y' and $y_{i,j}$ are right-circulant 18×133 cells for the next 28 rows.

Table 6. The coefficients of $a(x) = a_0x^0 + a_1x^1 + \cdots + a_{132}x^{132}$ in $\mathbb{F}_2[x]/(x^{133} - 1)$

Pol.	$(a_0, a_1, a_2, \ldots, a_{132})$
$e_1(x)$	1110100111010011101001110100111010011101001110100111010011101001110 100111010011101001110100111010011101001110100111010011101001110100
$\alpha_1(x)$	1001110100111010011101001110100111010011101001110100111010011101001 110100111010011101001110100111010011101001110100111010011101001110
$e_2(x)$	1001011100101110010111001011100101110010111001011100101110010111001 011100101110010111001011100101110010111001011100101110010111001011
$\alpha_2(x)$	1011100101110010111001011100101110010111001011100101110010111001011 100101110010111001011100101110010111001011100101110010111001011100
$e_3(x)$	0001001101011110011001111010100100111001011111111101100110010111000 010111101011000101011111110101110001111000011100100110010101000000
$\alpha_3(x)$	0111110001010000001101110010011110001111101111011111110111100010101 101100010010101100100001100000101010000100110011001001100101111010
$e_4(x)$	0000000101010011001001110000111100011101011111111010100011010111010 000111010011001101111111110100111001001010111100110011110101100100
$\alpha_4(x)$	0010111101001100100110011001000010101000000110000100110101001000110 1 101010001111011111111011110111111000111100100111011000000010100011111
$e_5(x)$	0111101110001111111000100111011101011010000100000111011001011100111 0 110100111010101001100000101001011100111110100110010111101001110100
$\alpha_5(x)$	1001110111001101110101010101001010011110100001000000100100001111001 1 001001010100101100101110110100110111010110010010000101110010000011
$e_6(x)$	0001011100101111010011001011111001110100101000001100101010111001011 011100111010011011100000010000101101011101110010001111111000111011 11
$\alpha_6(x)$	1110000001001110100001001001101011101100101101110100110100101010010 0 110011110000100100000010000101111001010010101010101110110011101110 0
$e_7(x)$	0110110110100010100011000101100111010000111001010110011110000111101 101100000001011011100101110110011011000110101111010101001111101101 1
$\alpha_7(x)$	1010100000101000000010011101110011110111010001111001111100010010011 11 001110110011111000000000111111010100010100011101110111111001111001
$e_8(x)$	0110110111111001010101111010110001101100110111010011101101000000110 1 101111000011110011010100111000010111001101000110001010001011011011
$\alpha_8(x)$	110011110011111101110111000101000101011111110000000001111100110111 0 011110010010001111100111100010111011110011101110010000000101000010 1
$e_9(x)$	0111111111111111111101111111111111111111110111111111111111111111011111111 1 11111111101111111111111111111111101111111111111111111111011111111111111111 1
$\alpha_9(x)$	1011011000010100010101101100001010001010110110000101000101011011000 010100010101101100001010001010110110000101000101011011000010100010