Vulnerabilities of "McEliece in the World of Escher"

Dustin Moody[1] and Ray Perlner[1]

[1]National Institute of Standards and Technology,
Gaithersburg, Maryland, USA

dustin.moody@nist.gov, ray.perlner@nist.gov

Abstract. Recently, Gligoroski et al. proposed code-based encryption and signature schemes using list decoding, blockwise triangular private keys, and a nonuniform error pattern based on "generalized error sets." The general approach was referred to as *McEliece in the World of Escher*. This paper demonstrates attacks which are significantly cheaper than the claimed security level of the parameters given by Gligoroski et al. We implemented an attack on the proposed 80-bit parameters which was able to recover private keys for both encryption and signatures in approximately 2 hours on a single laptop. We further find that increasing the parameters to avoid our attack will require parameters to grow by (at least) two orders of magnitude for encryption, and may not be achievable at all for signatures.

Keywords: Information Set Decoding, Code-based Cryptography, McEliece PKC, McEliece in the World of Escher

1 Introduction

The McEliece cryptosystem [1] is one of the oldest and most studied candidates for a postquantum cryptosystem. McEliece's original scheme used Goppa codes, but other families of codes have been proposed, such as moderate density parity check codes [2] and low rank parity check codes [3, 4]. Recently, Gligoroski et al. [5, 6] proposed a new approach to designing a code-based cryptosystem. Their approach uses a blockwise-triangular private key to enable decryption and signatures through a list decoding algorithm. The error vector in both cases is characterized, not by a maximum Hamming weight t, as is typical for code-based cryptosystems, but by an alphabet of allowed ℓ-bit substrings known as the *generalized error set*. Claimed advantages of this approach include a straightforward signature scheme and the ability to analyze security by using the tools of algebraic cryptanalysis.

The concept of information set decoding originates with Prange [7]. Further optimizations were subsequently proposed by Lee and Brickell [8], Leon [9], Stern [10], and several others [11–14]. Information set decoding techniques can be used to attack code-based cryptosystems in several ways. They can be used to search for a low-weight error vector directly, or they can be used to detect

Disclaimer: Any mention of commercial products or organizations is for informational purposes only; it is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the products identified are necessarily the best available for the purpose.

hidden structure in the public generator or parity check matrices by finding low weight code words in the row space of the generator matrix or parity check matrix. All of these applications of information set decoding are relevant to the scheme of Gligoroski et al. We will refer to their scheme as *McEliece Escher*, since it was introduced in their paper *McEliece in the World of Escher* [5, 6]. We demonstrate that information set decoding techniques are much more effective against the McEliece Escher scheme than suggested by the authors' original security analysis.

Gligoroski et al. were aware of both categories of information set decoding attacks on their scheme, but their analysis of these attacks was incomplete. Most seriously, they believed that information set decoding only produced a distinguisher on the private key, rather than a full key recovery, and they failed to consider the application of information set decoding to find a valid error vector in the signature setting. Landais and Tillich [15] applied similar techniques to convolutional codes, which have similar structure to the private keys used by McEliece Escher. We offer improvements to the existing approaches, including showing how to take advantage of the structured permutation used by McEliece Escher to disguise the private generator matrix.

Furthermore, we show our attacks are practical. Using the proposed parameters for 80-bits of security, we were able to recover private keys for both encryption and signatures in less than 2 hours on a single laptop. We find that increasing the parameters to avoid our attack will require parameters to grow by (at least) two orders of magnitude for encryption, and may not be practical at all for signature.

2 Background: McEliece schemes

2.1 Public and Private Keys

Gligoroski et al. construct their scheme along the lines of the original McEliece cryptosystem. The public key is a $k \times n$ generator matrix G_{pub} for a linear code over $\mathbb{F}_2$. To encrypt a message, the sender encodes a k-bit message m as an n bit codeword and then intentionally introduces errors by adding an error vector e. The ciphertext is then given by:

$$c = mG_{pub} + e.$$

Gligoroski et al. also introduce a signature scheme by applying the decoding algorithm to a hashed message. A signature σ is verified by checking

$$\mathcal{H}(m) = \sigma G_{pub} + e,$$

for a suitably chosen hash function $\mathcal{H}$.

Similar to the ordinary McEliece scheme, G_{pub} is constructed from a structured private generator matrix G, an arbitrary $k \times k$ invertible matrix S, and an $n \times n$ permutation matrix P.

$$G_{pub} = SGP. \tag{1}$$

For encryption, G_{pub} must be chosen in such a way that the private key allows unique decoding of a properly constructed ciphertext. For signatures, on the other hand, G_{pub} must be constructed to allow some decoding (not necessarily unique) of a randomly chosen message digest.

It will sometimes be helpful to characterize the public and private codes by their parity check matrices. The private parity check matrix, H is a $(n-k) \times n$ matrix, related to the private generator matrix G by the relation

$$GH^T = 0.$$

Similarly, it is easy to construct a public parity check matrix H_{pub} from G_{pub}, characterized by the relation $G_{pub}H_{pub}^T = 0$. This will be related to the private parity check matrix as

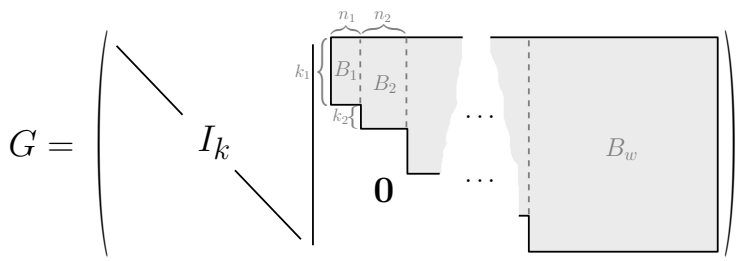
$$H_{pub} = S'HP,$$

where S' is an $(n-k) \times (n-k)$ invertible matrix and P is the same permutation matrix as in Equation (1).

2.2 Private Generator and Parity Check Matrices

To construct the binary (n, k) code used in the McEliece Escher scheme, the (private) generator matrix is of the form illustrated in Figure 1. Each block B_i

Fig. 1. The private generator matrix

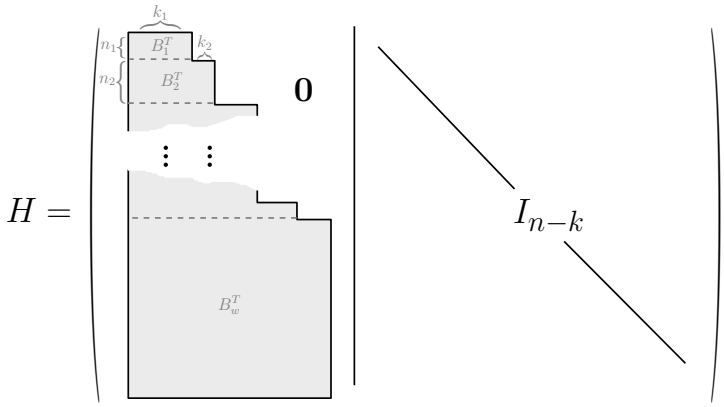


is a random binary matrix of dimension $(\sum_{j=1}^{i} k_j) \times n_i$, so that $k = k_1 + k_2 + \cdots + k_w$ and $n = k + n_1 + n_2 + \cdots + n_w$. The corresponding private parity check matrix is depicted in Figure 2, and has a similar block-wise structure. For ease of notation, we will let $K = (k_1, k_2, .., k_w)$ and $N = (n_1, n_2, .., n_w)$.

2.3 Error Sets

In the McEliece Escher scheme, the error vector is broken up into n/ℓ segments, each ℓ-bits. The value ℓ is called the *granularity* of the scheme, and for all proposed parameter sets, ℓ is set to 2. While the original McEliece scheme restricted the error vectors to having a low Hamming weight t, the McEliece

Fig. 2. The private parity check matrix



Escher scheme instead restricts the error space by choosing each ℓ-bit subsegment from a limited alphabet, called an *error set*. Error sets may be analyzed in terms of a density parameter ρ given by the formula

$$\rho = |E|^{1/\ell}.$$

For the proposed parameters, the error set is always $E = \{00, 01, 10\}$. This error set has granularity $\ell = 2$ and density $\rho = \sqrt{3}$.

Since public key operations require the encrypter or verifier to distinguish between valid and invalid error vectors, the permutation P used to disguise the private generator and parity check matrices must necessarily be of a special form. The action of P needs to rearrange ℓ-bit segments of the rows, but leave the segments themselves intact. In other words, P must consist of $\ell \times \ell$ blocks which are either 0 or the identity matrix I_ℓ.

3 Improving Information Set Decoding for the Error Vector

Information set decoding may be used to recover m and e from the ciphertext $c = mG_{pub} + e$. The basic strategy involves guessing k bits of the error vector and recovering the rest by linear algebra. One of the simplest information set decoding algorithms is given in Algorithm 1.

It should be clear that the number of iterations this algorithm requires is inversely proportional to the probability that an attacker can guess k bits of the error vector. As in the case of standard McEliece, the most probable guess for these k bits is the all zero vector. However, since McEliece Escher uses a nonuniform error pattern, the choice of the permutation P' has a significant effect on the probability of success. In their security analysis, Gligoroski et al. assumed that P' would be of similar form to the secret permutation matrix P used to disguise the private key. This has the effect of forcing the adversary to guess all the bits in each ℓ-bit block chosen from a generalized error

Algorithm 1: Information set decoding for the error vector

Input: ciphertext c, and a parameter k
Output: message m, error e

1. Permute the bits of the ciphertext by a random permutation matrix P':

$$\begin{aligned} c' &= (mG_{pub} + e)P' \\ &= mG_{pub}P' + eP' \\ &= m(A|B) + (e'_1|e'_2) \\ &= (mA + e'_1)|(mB + e'_2), \end{aligned}$$

   where A and e'_1 are the first k columns of the permuted generator matrix $G_{pub}P'$ and permuted error vector eP', respectively.
2. If A is not invertible, go to step 1.
3. Guess e'_1. If correct the message can be reconstructed as

$$m = ((mA + e'_1) - e'_1)A^{-1}.$$

   The error vector is then $e = c - mG_{pub}$.
4. If the error vector is properly formed (i.e., the Hamming weight is less than t for standard McEliece, or composed of ℓ-bit substrings from the proper generalized error set in McEliece Escher), return m and e. Otherwise go back to step 1 and start over with a new permutation P'.

set. Thus the probability of each guess is ρ^{-k}. However, an attacker can do better by choosing a permutation that always separates the bits of an ℓ-bit block. For example, each bit is 0 two-thirds of the time when the error set is $E = \{00, 01, 10\}$, but both bits are 0 only one-third of the time. By guessing one bit within each 2-bit block, an attacker achieves a success probability of $(2/3)^k$, which is a significant improvement over the value $(1/\sqrt{3})^k$ assumed by Gligoroski et al.'s security analysis. Concretely, when used against Gligoroski et al.'s claimed 80-bit secure code with parameters $(n, k) = (1160, 160)$, the probability of a single guess of k bits of the error vector improves from 2^{-127} to 2^{-94}.

Similar improvements are available for more sophisticated decoding algorithms. In section 5.1 of their paper [5], Gligoroski et al analyze modifications to several information set decoding algorithms [8, 10–14], including several that use meet-in-the-middle strategies to try several guesses at once, and apply them to the case where $k = 256$. For our purposes these algorithms may be characterized by the number of bits $k+\lambda$ which are guessed, along with the Hamming weight p of those guesses. Whenever $p \cdot \log_2(\sqrt{3}) < (k + \lambda) \log_2(\frac{2}{\sqrt{3}})$, the modification described above decreases the complexity of decoding by a factor of at least $2^{(k+\lambda)\log_2(\frac{2}{\sqrt{3}}) - p \cdot \log_2(\sqrt{3})}$. This is true for some of the algorithms analyzed by Gligoroski et al. For example, Stern's algorithm is quoted as having a complexity of 2^{197} when applied to $k = 256$, however, with our modification, Stern's algorithm with $p = 2$ has a probability of success per iteration of approximately 2^{-136} corresponding to a complexity somewhere around 2^{150}. It does not however appear that a direct application of our modification improves

the most efficient algorithm analyzed by Gligoroski et al., since p is apparently too large. This algorithm, adapted from the BJMM algorithm [14], is quoted as achieving a complexity of 2^{123}. It is possible that some sort of hybrid approach will provide an improvement. Nonetheless, for the remainder of this paper, we will assume that Gligoroski et al.'s analysis of the complexity of attacking the encryption algorithm, by direct search for a unique patterned error vector, is correct.

Algorithm 1, modified so that as many ℓ-bit blocks as possible of the error are spit between e'_1 and e'_2, is however an extremely effective method for signature forgery. For the error set $E = \{00, 01, 10\}$, when a 2-bit block is split between e'_1 and e'_2, the bit in e'_1 may be forced to 0, and the pair of bits will remain within the error set, whether the corresponding bit in e'_2 is set to 0 or 1. If all the bits of e'_1 are set to 0, then the probability for the resultant error vector e to be a valid error vector is $(\frac{\sqrt{3}}{2})^{n-2k}$. For the claimed 80-bit secure signature code with parameters $(n, k) = (650, 306)$, this probability is approximately 2^{-8}.

4 Information Set Decoding for the Private Key

Information set decoding techniques can also be used to find low weight elements in the row spaces of matrices. In our case, we are interested in the public generator and parity check matrices, G_{pub} and H_{pub}. Note that elements of these public row spaces are related to the elements of the row spaces of the private generator and parity check matrices by the permutation P used in the construction of the public key:

$$vG_{pub} = ((vS)G)P,$$

$$v'H_{pub} = ((v'S')H)P,$$

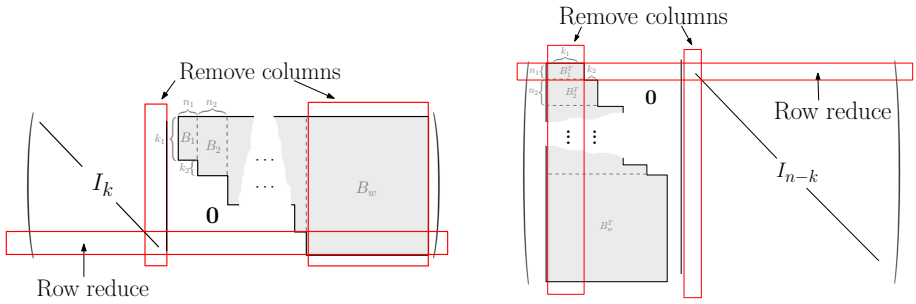
where v and v' are k and $(n-k)$-bit row vectors respectively. Consequently, the result of an information set decoding attack on G_{pub} or H_{pub} will simply be the image under P of a low weight element of the row space of G or H. We thus examine the space of low weight vectors for encryption and signatures.

Recall the description of the private generator and parity check matrices given in Section 2.2. For encryption, the private key operation requires maintaining a list of at least ρ^{k_1} entries. This means that k_1 must be small in order for the scheme to be efficient. The first n_1 rows of H are forced by construction to have nonzero bits only in the $(n_1 + k_1)$ columns $C_j(H)$, with $1 \leq j \leq k_1$ or $k + 1 \leq j \leq k + n_1$. Linear combinations of these rows will then produce approximately $\binom{n_1+k_1}{t} 2^{-k_1}$ distinct row vectors of weight t. The general attack strategy will be to seek to sample from the images under P of this space of low weight row vectors, which are constrained to only contain nonzero bits in columns C_j, with the same bounds on j as above. We thereby learn the images of those columns, and once learned they can be removed from H_{pub}. The row space of the matrix formed by the remaining columns of H is the same as

for the parity check matrix of a code of the same structure with $w' = w - 1$, $N' = (n_2, .., n_w)$, $K' = (k_2, .., k_w)$. Applying this strategy recursively will allow us to identify the underlying block structure and construct a new private key of the same form.

For signatures, the private key operation requires maintaining a list of at least $(2/\rho)^{n_w}$ entries. In order for the scheme to be efficient, n_w must be small. The last k_w rows of G have zero bits everywhere, except possibly in the (k_w+n_w) columns $C_j(G)$, indexed by $(k-k_w+1) \leq j \leq k$ and $(n-n_w+1) \leq j \leq n$. Linear combinations of the rows will produce approximately $\binom{k_w+n_w}{t}2^{-n_w}$ distinct row vectors of weight t. Similarly as done for encryption, the strategy for signatures will be to seek to sample from the images under P of this space of low weight row vectors, learning the images of the aforementioned columns. Once the columns have been learned, they can be removed from G_{pub} and the process recursively repeated since the row space of the matrix formed by the remaining columns of G is that of a parity check matrix for a code of the same form with $w' = w - 1$, $N' = (n_1, .., n_{w-1})$, $K' = (k_1, .., k_{w-1})$. See Figure 3 for an illustration of the strategy for both encryption and signatures.

Fig. 3. Removing columns and row-reducing leaves a smaller code of the same form.



It should be noted that the space of short vectors with support on the target columns is not the only source of low weight vectors that can be obtained by information set decoding algorithms. However, for realistic parameters, it is generally advantageous to simply choose t to maximize the rate at which vectors from the target space are produced. This is because there is an efficient way to use a list of vectors, some of which are from the target space and some of which are not, to produce a full list of the target columns. The algorithm that does this uses a subroutine which is applied to a small subset of the list of vectors, and which will usually produce the full list of target columns if the chosen vectors are all from the target space. This subroutine will not only terminate quickly on correct inputs, but also if one of the vectors is not from the target space. In the latter case the algorithm will recognizably fail, by identifying too many columns. The first obtained list of vectors, required to recover the full target set of columns, will generally be small enough that trying the subroutine on all appropriately sized subsets of the list will be of insignificant cost compared to the information set decoding steps.

The subroutine proceeds as follows (see Alg. 2). The input is a list of target columns, containing at least $(k_1 + 1)$ of the target columns for encryption (or at least $(n_w + 1)$ of the target columns for signatures). These columns may generally be obtained by combining the nonzero positions of a small number (e.g. two) of the target vectors produced by an information set decoding algorithm, such as Stern's algorithm.

Algorithm 2: Subroutine to complete the list of target columns

Input: A set S of columns
Output: A set of columns S' $\supseteq$ S, and a flag "Success" or "Failure"

1. Check whether removing the columns of S from the public matrix reduces the rank.
   - If all of the columns are from the target set, then removing the columns in S will likely reduce the rank of the public matrix by $|S| - k_1$ for encryption (or $|S| - n_w$ for signatures).
2. For each column C not in S, check whether the rank of the public matrix is decreased when C is removed in addition to those already in S.
   (a) if the rank is decreased, add C to S and repeat step 2.
   (b) if the rank stays the same for each $C \notin S$, return $S' = S$ and go to the last step to determine success.
3. The algorithm succeeds if the rank stops decreasing at $n - k - n_1$ for encryption (or $k - k_w$ for signatures). Otherwise output failure.

4.1 Using the Nonrandom P

The attack outlined in the previous section does not take into account the constraints on the permutation P used to disguise the private key G (or H). In particular, the permutation leaves blocks of ℓ consecutive columns intact. Thus, there is additional information about the location of our target columns that we did not use. In particular, if the column C_j is in our target set, we can be confident that all the columns $C_{\lfloor \frac{j-1}{\ell} \rfloor + 1}, ..., C_{\lfloor \frac{j-1}{\ell} \rfloor + \ell}$ are also in the target set. We modify Stern's algorithm to take advantage of this by choosing our random permutation P' in such a way as to leave ℓ-bit blocks of columns intact, just as the private matrix P does. We will also count the number of nonzero ℓ-bit blocks within a row vector as a substitute for Hamming weight, wherever Hamming weight is used by Stern's algorithm. We will refer to this altered weight as *block-weight*. Taking into account the special form of P also has other beneficial effects for the attacker. In particular, Algorithm 2 has a higher probability of success when the rank effects of the inclusion of blocks of ℓ columns (instead of individual columns) are considered, since it is much less likely for these blocks to be totally linearly dependent on each other, for reasons other than the overall block structure of the matrix.

The modified version of Stern's algorithm proceeds as shown in Algorithm 3. Note the Stern's algorithm window size will be denoted L, instead of the standard l, to avoid confusion with the granularity.

Algorithm 3: Modified Stern's Algorithm

Input: a matrix G_{pub}, parameters p, t, L, ℓ

Output: a vector in the row space of G_{pub} which has block-weight t

1. Permute the columns of G_{pub} :

$$G'_{pub} = G_{pub}P',$$

where P' is a permutation matrix consisting of $\ell \times \ell$ blocks which are either zero or the identity, but otherwise chosen randomly.

2. Check that the first k columns of the new matrix G'_{pub} form an invertible matrix A. If A is not invertible, go back to step 1.
3. Left-multiply by A^{-1}, resulting in a matrix of the form

$$M = A^{-1}G'_{pub} = \left[I_k \mid Q\right].$$

4. Search for low-weight row-vectors among linear combinations involving small subsets of the rows of M:
   (a) Divide the rows of M into two equal length lists, i.e., for $0 < i \leq \frac{k}{2\ell}$, and for $B = (b_1, .., b_\ell) \in \mathbb{F}_2^\ell$

$$x_{i,B} = \sum_{r=1}^{\ell} b_r \mathrm{row}_{i\ell+r}(M).$$

   Similarly, for $\frac{k}{2\ell} < j \leq \frac{k}{\ell}$

$$y_{j,B} = \sum_{r=1}^{\ell} b_r \mathrm{row}_{j\ell+r}(M).$$

   (b) Compute each possible sum of all subsets of size p of the $x_{i,B}$, as well as for all possible sums of p of the $y_{j,B}$. Check for collisions on bits $(k+1), \ldots, (k+L)$:

$$\mathrm{bits}_{k+1,\ldots,k+L\ell}(x_{i_1,B_1} + \ldots + x_{i_p,B_p}) = \mathrm{bits}_{k+1,\ldots,k+L\ell}(y_{j_1,B_1} + \ldots + y_{j_p,B_p}).$$

   (c) When such a collision is found, compute the sum s of the $2p$ colliding row vectors

$$s = x_{i_1} + \ldots + x_{i_p} + y_{j_1} + \ldots + y_{j_p}.$$

   If the block-weight of any such s is equal to t return sP'. Otherwise, go back to step 1.

We now give an analysis of the complexity of obtaining the full list of target columns using this modified Stern's algorithm. Note that this analysis is only approximate, a tighter analysis may be possible using techniques similar to those outlined in section 5 of [16]. For each block-weight t target vector g, the search will succeed if and only if gP' has block-weight p on its first $\frac{k}{2}$ bits, block-weight p on the next $\frac{k}{2}$ bits, and block-weight 0 on the next L bits. For a randomly chosen P' this probability is

$$\mathrm{Prob}(n,k,p,\ell,L,t) = \binom{n/\ell}{t}^{-1}\binom{k/(2\ell)}{p}^{2}\binom{(n-k-L)/\ell}{t-2p},$$

and the equivalent probability for an attack on H_{pub} is

$$\mathrm{Prob}(n,n-k,p,\ell,L,t) = \binom{n/\ell}{t}^{-1}\binom{(n-k)/(2\ell)}{p}^{2}\binom{(k-L)/\ell}{t-2p}.$$

The approximate number $\mathcal{D}$ of distinct target vectors of a given weight t is

$$\mathcal{D}_{sig} \approx \binom{(k_w+n_w)/\ell}{t}\left(2^{\ell}-1\right)^{t}\cdot 2^{-n_w},$$

for signature, and for encryption

$$\mathcal{D}_{enc} \approx \binom{(n_1+k_1)/\ell}{t}\left(2^{\ell}-1\right)^{t}\cdot 2^{-k_1}.$$

The expected number $\mathcal{E}$ of target vectors required for a successful attack is

$$\mathcal{E}_{sig} \approx \left\lceil \frac{\log\left(\frac{k_w}{k_w+n_w}\right)}{\log\left(\frac{k_w+n_w-t\ell}{k_w+n_w}\right)} \right\rceil,$$

for signature, and for encryption

$$\mathcal{E}_{enc} \approx \left\lceil \frac{\log\left(\frac{n_1}{n_1+k_1}\right)}{\log\left(\frac{n_1+k_1-t\ell}{n_1+k_1}\right)} \right\rceil.$$

The total number of iterations of the modified Stern's algorithm is therefore

$$i_{sig} \approx \left\lceil \frac{\log(\frac{k_w}{k_w+n_w})}{\log(\frac{k_w+n_w-t\ell}{k_w+n_w})} \right\rceil \cdot \binom{(k_w+n_w)/\ell}{t}^{-1}\left(2^{\ell}-1\right)^{-t}2^{n_w} \cdot \binom{n/\ell}{t}\binom{k/(2\ell)}{p}^{-2}\binom{(n-k-L)/\ell}{t-2p}^{-1},$$

and

$$i_{enc} \approx \left\lceil \frac{\log(\frac{n_1}{n_1+k_1})}{\log(\frac{n_1+k_1-t\ell}{n_1+k_1})} \right\rceil \cdot \binom{(n_1+k_1)/\ell}{t}^{-1}\left(2^{\ell}-1\right)^{-t}2^{k_1} \cdot \binom{n/\ell}{t}\binom{(n-k)/(2\ell)}{p}^{-2}\binom{(k-L)/\ell}{t-2p}^{-1}.$$

5 Experimental Results

We implemented the attacks described in the previous section on a standard laptop with a 2.2 GHZ Intel core i7 processor. We used the parameters suggested by Gligoroski et al. for 80 bits of security. Concretely, for encryption $n = 1160, k = 160, \ell = 2, w = 17$, with $K = (32, 8, 8, ..., 8)$ and $N = (32, 32, ..., 32, 488)$. We used parameters $(t, p, L) = (11, 1, 9)$ for the modified Stern's algorithm, which needed approximately 1000 iterations in our trials. The predicted value from the analysis in the previous section was 2500. The total wall time for the computation to recover a private key was on average less than 2 hours.

For signatures, we have $n = 650, k = 306, \ell = 2, w = 6$, with $K = (84, 48, 48, 48, 48, 30)$ and $N = (48, 48, 48, 48, 48, 104)$. The modified Stern parameters we used were $(t, p, L) = (40, 1, 7)$. With such a high value for t, a higher number of iterations were needed, usually less than 10000 (the predicted value was around 4900). The total wall time was again less than 2 hours on average.

6 Countermeasures

Attempts to increase the security of McEliece Escher by altering the parameters are severely constrained by the requirement that ρ^{k_1} be small for encryption and that $(2/\rho)^{n_w}$ be small for signatures.

One possiblility would be to try to decrease ρ (or $2/\rho$), as appropriate, to allow k_1 or n_w to increase. This, however, turns out to be counterproductive. Due to the attack in Section 4.1, we see what really matters for security is that k_1/ℓ be large for encryption, or n_w/ℓ be large for signatures. Asymptotically, there will be 2^ℓ vectors in the row space of H_{pub} of block-weight no more than $k_1/\ell + 1$ and 2^ℓ vectors in the row space of G_{pub} of block-weight no more than $n_w/\ell + 1$. The factor of 2^ℓ will make up for the increased cost per iteration of the modified Stern's algorithm with $p = 1$, but the probability of success per iteration will remain at approximately $(\frac{k}{n})^{k_1/\ell}$ for encryption and $(\frac{n-k}{n})^{n_w/\ell}$ for signatures. Encryption requires $(\rho^\ell)^{k_1/\ell}$ to be small for efficiency and k_1/ℓ to be large for security. Thus the ideal value for ρ and ℓ would minimize ρ^ℓ. Likewise, the signature scheme requires $((\frac{2}{\rho})^\ell)^{n_w/\ell}$ to be small for efficiency and n_w/ℓ to be large for security. Hence, the ideal value for ρ and ℓ would minimize $(\frac{2}{\rho})^\ell$.

While it is possible to decrease ρ (or $\frac{2}{\rho}$) by increasing ℓ, the consequence is that ρ^ℓ and $(\frac{2}{\rho})^\ell$ both increase at least linearly in ℓ for error sets of the proper form (for security, the generalized error set cannot impose linear constraints on the error vector, e.g. by forcing a bit of the error vector to always be 0). Thus, fixing $\frac{n}{k}$ and the security level, we find that the cost of decryption increases when we increase ℓ.

A better idea is to greatly increase n_w for encryption and k_w for signatures. This works by making $\frac{k}{n}$ very small for encryption and $\frac{n-k}{n}$ very small for signatures. In the context of an information set decoding attack, this has the

effect of decreasing the probability that a given nonzero bit (or ℓ-bit block) of a target vector will be placed outside the information set by a randomly chosen (block) permutation. This is a much better solution for signatures than for encryption. For typical parameters, the modified Stern's algorithm requires ~ 30 nonzero blocks to fall outside the information set when attacking a signature. Thus, bringing the cost of the attack from $\sim 2^{30}$ to $\sim 2^{80}$ should only require $\frac{n-k}{n}$ to fall from about 0.5 to about 0.15. That is, the size of the 80-bit-secure code increases from a 650×304 bit generator matrix to a 2000×1654 bit generator matrix. For attacking typical encryption parameters, the modified Stern's algorithm only requires ~ 6 nonzero blocks to fall outside the information set. This means $\frac{k}{n}$ needs to fall from about 0.15 to 0.0005. The result is that for an 80-bit-secure code, the size would increase from 1160×160 to $300,000 \times 160$.

There is however an additional complication created by the above countermeasure for signatures. A code with error set $E = \{00, 01, 10\}$ can be trivially broken whenever $\frac{n-k}{n} < 0.5$ due to the attack described at the end of Section 3. This attack may be generalized to apply to other error sets, whenever there is a linear projection from $\mathbb{F}_2^\ell \to \mathbb{F}_2^{\ell'}$ with $\ell' \leq \frac{k}{n}\ell$ such that an element of $\mathbb{F}_2^\ell$ with a certain fixed projection onto $\mathbb{F}_2^{\ell'}$ is a member of the error set with very high probability. Thus in order to avoid attack, the error set must be chosen so that there is no such projection. We have not found any way to do this that makes the honest party's signing operation (list decoding for signatures) asymptotically more efficient than both attacks (ISD for the error vector and ISD for the private key.)

7 Conclusion

We demonstrate practical attacks on the proposed parameters of McEliece Escher. The poor choice of parameters is a demonstration of the general principle that code-based schemes should be designed in such a way as to avoid all practical distinguishers on the public key, since distinguishers can often be modified, at little cost, to create private-key recovery attacks. Additionally, our cryptanalysis demonstrates that information set decoding techniques can be modified to take advantage of code-based schemes whose private keys are disguised by a structured, rather than a completely random, permutation matrix. The recent cryptanalysis of cyclosymmetric-MDPC McEliece by Perlner [17] is another example of this general principle. This technique is especially effective in creating signature forgeries.

For encryption, it appears the above pitfalls can be compensated for, by simply making the parameters of McEliece Escher larger. However, this requires making the keys at least two orders of magnitude larger. This is a major burden on an already inefficient scheme. Asymptotically, these modifications can only make the complexity of a key-recovery attack quasi-polynomially worse than the complexity of decryption by the honest party.

References

1. McEliece, R.J.: A Public-Key Cryptosystem Based On Algebraic Coding Theory. Deep Space Network Progress Report **44** (1978) 114–116
2. Misoczki, R., Tillich, J.P., Sendrier, N., Barreto, P.S.L.M.: MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes. Cryptology ePrint Archive, Report 2012/409 (2012) `http://eprint.iacr.org/`.
3. Gaborit, P., Murat, G., Ruatta, O., Zemor, G.: Low Rank Parity Check codes and their application to cryptography. In Lilya Budaghyan, Tor Helleseth, M.G.P., ed.: The International Workshop on Coding and Cryptography (WCC 13), Bergen, Norway (2013) 13 p. ISBN 978-82-308-2269-2.
4. Gaborit, P., Ruatta, O., Schrek, J., Zmor, G.: RankSign: An Efficient Signature Algorithm Based on the Rank Metric. In Mosca, M., ed.: Post-Quantum Cryptography. Volume 8772 of Lecture Notes in Computer Science. Springer International Publishing (2014) 88–107
5. Gligoroski, D., Samardjiska, S., Jacobsen, H., Bezzateev, S.: McEliece in the World of Escher. Cryptology ePrint Archive, Report 2014/360 (2014) `http://eprint.iacr.org/`.
6. Gligoroski, D.: A New Code Based Public Key Encryption and Signature Scheme Based on List Decoding. (Presented at "Workshop on Cybersecurity in a Post-Quantum World," NIST, Gaithersburg MD, USA)
7. Prange, E.: The Use of Information Sets in Decoding Cyclic Codes. Information Theory, IRE Transactions on **8** (1962) 5–9
8. Lee, P., Brickell, E.: 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., Seegmller, G., Stoer, J., Wirth, N., Gnther, C., eds.: Advances in Cryptology EUROCRYPT 88. Volume 330 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1988) 275–280
9. Leon, J.: A Probabilistic Algorithm for Computing Minimum Weights of Large Error-correcting Codes. Information Theory, IEEE Transactions on **34** (1988) 1354–1359
10. Stern, J.: A Method for Finding Codewords of Small Weight. In Cohen, G., Wolfmann, J., eds.: Coding Theory and Applications. Volume 388 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1989) 106–113
11. Finiasz, M., Sendrier, N.: Security Bounds for the Design of Code-Based Cryptosystems. In Matsui, M., ed.: Advances in Cryptology ASIACRYPT 2009. Volume 5912 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2009) 88–105
12. Bernstein, D., Lange, T., Peters, C.: Smaller Decoding Exponents: Ball-Collision Decoding. In Rogaway, P., ed.: Advances in Cryptology CRYPTO 2011. Volume 6841 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 743–760
13. May, A., Meurer, A., Thomae, E.: Decoding Random Linear Codes in $\tilde{O}(2^{0.054n})$. In Lee, D., Wang, X., eds.: Advances in Cryptology ASIACRYPT 2011. Volume 7073 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 107–124
14. Becker, A., Joux, A., May, A., Meurer, A.: Decoding Random Binary Linear Codes in $2^{n/20}$: How $1+1=0$ Improves Information Set Decoding. In Pointcheval, D., Johansson, T., eds.: Advances in Cryptology EUROCRYPT 2012. Volume 7237 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 520–536
15. Landais, G., Tillich, J.P.: An Efficient Attack of a McEliece Cryptosystem Variant Based on Convolutional Codes. In: Post-Quantum Cryptography. Springer (2013) 102–117
16. Otmani, A., Tillich, J.P.: An efficient attack on all concrete kks proposals. In Yang, B.Y., ed.: Post-Quantum Cryptography. Volume 7071 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 98–116
17. Perlner, R.: Optimizing Information Set Decoding Algorithms to Attack Cyclosymmetric MDPC Codes. In Mosca, M., ed.: Post-Quantum Cryptography. Volume 8772 of Lecture Notes in Computer Science. Springer International Publishing (2014) 220–228