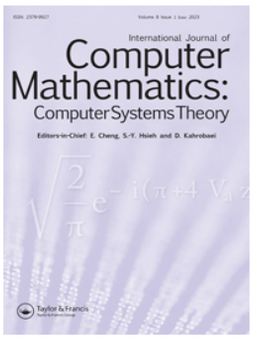

International Journal of Computer Mathematics: Computer Systems Theory

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tcom20

A novel attack on McEliece's cryptosystem

Henry Gray, Christopher Battarbee, Siamak F. Shahandashti & Delaram Kahrobaei

To cite this article: Henry Gray, Christopher Battarbee, Siamak F. Shahandashti & Delaram Kahrobaei (2023): A novel attack on McEliece's cryptosystem, International Journal of Computer Mathematics: Computer Systems Theory, DOI: 10.1080/23799927.2023.2229278

To link to this article: https://doi.org/10.1080/23799927.2023.2229278

Published online: 11 Jul 2023.

INTERNATIONAL JOURNAL OF COMPUTER MATHEMATICS: COMPUTER SYSTEMS THEORY
https://doi.org/10.1080/23799927.2023.2229278

A novel attack on McEliece's cryptosystem

Henry Gray, Christopher Battarbee, Siamak F. Shahandashti and Delaram Kahrobaei

University of York, York, UK

ABSTRACT

This report proposes a new and novel attack on McEliece's cryptosystem that improves on the probability of attacks formerly proposed by Stern, and Lee and Brickell.

Modern day encryption standards have been long since proven insecure to quantum attack, and quantum-resistant cryptosystems are now at the forefront of research. Since 2016, the National Institute of Standards and Technology (NIST) has presided over a public competition to establish new standards for public-key encryption that will secure our data in the post-quantum world. Now in its final round, one of the remaining candidates is McEliece's cryptosystem, a code-based cryptosystem proposed in 1978 by Robert J. McEliece. With a few minor alterations since its conception, McEliece's cryptosystem has, so far, proven resistant to quantum attacks, making it an ideal finalist candidate. The cryptosystem has not, however, escaped the attention of attack and, over the last four decades, a variety of algorithms have been proposed with the intention of exploiting it to recover the plaintext.

This paper initially provides an overview of McEliece's cryptosystem and two existing attacks proposed by Stern, and Lee and Brickell in the 1980s. Observations are made on the shared probabilistic nature of Stern's algorithm, and Lee and Brickell's attack. It is noted that the first step of both algorithms involves the random selection of a subset of n indexes. In Stern's algorithm, $n-k$ of n columns in a matrix H are chosen at random and, in Lee and Brickell's attack, k of n bits of the ciphertext are selected, also at random. This relationship is exploited to compound the two attacks and propose a new, novel attack. The complexity and probability of the new attack are discussed and an analysis is conducted to compare it against both Stern's algorithm and Lee and Brickell's attack.

This analysis suggests that the probability of successful attack comes close to combining those of the two original attacks. Furthermore, the results suggest that the novel attack can successfully recover a message faster than Stern's algorithm. Improvements to the attack are suggested, concluding that further study should be conducted into fully analysing it and its implications on the security of McEliece's cryptosystem.

ARTICLE HISTORY
Received 18 October 2022
Revised 15 March 2023
Accepted 7 June 2023

KEYWORDS
Code-based cryptography; McEliece's cryptosystem

1. Introduction

1.1. Post-quantum cryptography

The emergence of quantum computers over the past half-century poses a frighteningly realistic threat to current day cryptography of which, in spite of the severe ramifications, many are unaware. Modern day encryption standards have existed since the 1970s with Diffie and Hellman introducing the

CONTACT Delaram Kahrobaei dkahrobaei@gc.cuny.edu

concept of public-key encryption in 1976 [6], and Rivest, Shamir and Adleman exploiting number theory to create the famous RSA cryptosystem in 1978 [15]. Typically, cryptosystems such as RSA rely on the intractability of a number theoretic problem for which it is *computationally infeasible* to solve, such as factorizing a large prime number into its two prime factors. Problems such as these are considered unsolvable on classical computers, therefore making them suitable candidates for the basis of cryptosystems as they cannot be cracked by conventional methods. However, quantum algorithms solving these problems have existed as early as the 1990s, with Shor providing a polynomial-time algorithm in 1994 for solving both prime factorization and the discrete logarithm problem on a quantum computer [16]. The potential consequence of this is insurmountable; being that, at the advent of widespread quantum computers, all cryptographic standards currently unbreakable by classical computers will be rendered obsolete.

In 2016, the National Institute of Standards and Technology (NIST) produced a report surmizing that the aforementioned advent of quantum computers may be reached in as little as 20 years [13]. The report defined post-quantum cryptography and its goal to be *'the [development] of cryptographic systems that are secure against both quantum and classical computers, and can interoperate with existing communications protocols and networks'* and called upon cryptographers and academics to design quantum-resistant cryptosystems that are resilient to attack by quantum algorithms, such as Shor's or Grover's [8]. These were to be submitted into a public competition to decide upon the new standards for public-key cryptography that will secure our information in the rapidly approaching post-quantum world.

1.2. McEliece's cryptosystem

Code-based cryptography also finds its origins in the 1970s when, in 1978, Robert J. McEliece proposed *a public-key cryptosystem based on algebraic coding theory* [11]. Unlike RSA and other public-key cryptosystems which derive from number theory, McEliece's proposition utilized coding theory and the hard problem of decoding a linear code to which errors have been added.

Coding theory itself was initially developed for the purpose of communication over unreliable channels, particularly with respect to error-correction. However, in the context of code-based cryptography, errors are not of concern, but rather privacy. In this way, McEliece utilized coding theory in the context of a reliable, but insecure channel where errors can be artificially inserted during encryption without affecting communication.

Almost half a century after being proposed, McEliece's cryptosystem still remains secure and unbroken, by both conventional and quantum attacks. Whilst a series of possible attacks have been devised [2–5,10,17] , their exploitations target the parameters chosen to construct the code, rather than exploiting the nature of the algorithm itself. For this reason, an adapted version of McEliece's original cryptosystem (dubbed *Classic McEliece*) stands as a finalist candidate in the fourth round of NIST's *post-quantum cryptography standardization process* [14], alongside two other code-based cryptosystems, BIKE and HQC, which stand as alternative candidates.

2. McEliece's cryptosystem

The following sections rely on background knowledge in some underlying concepts of coding theory and cryptography, particularly error-correcting codes and public key cryptography. Menezes et al. provide a comprehensive introduction to cryptography in their book [12].

McEliece's cryptosystem also relies on binary Goppa Codes, a class of error correcting code, devised in 1970, by Valerii Densovich Goppa (in Russian) [1,7].

They are constructed from the following components [9]:

- *'The Support'*: a list $L = (a_1, \ldots a_n)$ of n distinct elements in $\mathbb{F}_q$
- *'The Goppa Polynomial'*: a polynomial $g(x) \in \mathbb{F}_q[x]$ of degree t satisfying the following properties:

- $g(x)$ is irreducible over $\mathbb{F}_q$, i.e. cannot be broken down into two (or more) composite factors
- $g(a) \neq 0$ for all $a \in L$.

The corresponding binary Goppa code $\Gamma(L, g)$ is defined by

$$\{\mathbf{c} \in \mathbb{F}_2^n | S(\mathbf{c}) = \sum_{i=1}^{n} \frac{c_i}{x - a_i} \equiv 0 \mod g(x)\}. \tag{1}$$

This, therefore, produces a linear code of length n. The code is able to correct up to t errors and has a minimum hamming distance of $d = 2t + 1$. The relationship between errors and minimum distance is therefore $t = \frac{d-1}{2}$.

Codewords can also be produced by a $k \times n$ generator matrix G. The parameters n and k also correspond to the lengths of the plaintext message and their codewords respectively, and the exact values of n and k are variable on the specific construction of the Goppa code. The codeword $\mathbf{c}$ corresponding to message $\mathbf{m}$ is obtained from the product of the message vector $\mathbf{m}$ and the generator matrix G

$$\mathbf{c} = \mathbf{m}G$$

A code C also has, associated with it, the $n \times (n - k)$ parity check matrix H. If a codeword $\mathbf{c}$ belongs to code C, the product of $\mathbf{c}H$, known as the *syndrome*, is a vector of length $(n - k)$ with hamming weight 0. However, if a codeword $\mathbf{c}'$ does not belong to the code C, the hamming weight of the syndrome vector will be greater than 0. Furthermore, the product of GH is always a matrix of 0s.

These details expanded upon further in the literature [1,7,9].

2.1. Key generation and encryption

McEliece's cryptosystem relies on a binary irreducible Goppa code C with the ability to correct up to t errors. The code C can be derived from the $k \times n$ generator matrix G. The public key is a tuple $(\hat{G}, t)$ consisting of a random permutation $\hat{G}$ of the aforementioned generator matrix and a value $t \in \mathbb{N}$. The code associated with the permuted generator matrix $\hat{G}$ has the same parameters and error correcting capability as the original code. The permuted generator matrix $\hat{G}$ is computed from the product of a random binary $k \times k$ non-singular matrix S and a random $n \times n$ permutation matrix P.

$$\hat{G} = SGP$$

To obtain the ciphertext $\mathbf{c}'$ from a binary message vector $\mathbf{m}$ of length k, we compute

$$\mathbf{c}' = \mathbf{m}\hat{G} + \mathbf{z}$$

where $\mathbf{z}$ is a random binary error vector of weight t and length n. The message $\mathbf{m}$ is therefore encrypted to a binary vector $\mathbf{c}'$ of length n. The decryption of $\mathbf{c}'$ to $\mathbf{m}$ is only possible with knowledge of the matrices S and P along with the error-correction algorithm for the code C, generated by G. It is therefore these values from which the private key is composed (S, G, P) (Or alternatively (S, D_G, P) where D_G is the decoding algorithm for C). Note that $\mathbf{c}'$ is used to denote the ciphertext as, in this paper, $\mathbf{c}$ denotes the codewords belonging to the Goppa code C.

2.2. Decryption

An n-length vector $\hat{\mathbf{c}}$ is computed from the product of the ciphertext $\mathbf{c}'$ and P^{-1}, the inverse of the matrix P.

$$\hat{\mathbf{c}} = \mathbf{c}'P^{-1}$$

As $\hat{\mathbf{c}} = \mathbf{c}'P^{-1} = (\mathbf{m}G + \mathbf{z})P^{-1} = (\mathbf{m}SGP + \mathbf{z})P^{-1} = (\mathbf{m}S)G + \mathbf{z}P^{-1}$, and $\mathbf{z}P^{-1}$ is a vector with weight t (i.e. a permutation of the error vector $\mathbf{z}$), the error-correction algorithm for the code C

Algorithm 1: McEliece's Cryptosystem

Key Generation: With parameters $n, t \in \mathbb{N}$, where $t \ll n$,
Generate the following matrices:
- G : $k \times n$ generator matrix for Goppa code C able to correct up to t errors and with minimum distance $d = 2t + 1$
- S : $k \times k$ random binary non-singular matrix
- P : $n \times n$ random permutation matrix

Compute the $k \times n$ matrix $\hat{G} = SGP$
Public Key: $(\hat{G}, t)$
Private Key: (S, G, P)
(Or (S, D_G, P), where D_G is an efficient decoding algorithm for G)
Encryption: To encrypt a binary message $\mathbf{m}$ of length k to ciphertext $\mathbf{c}'$ of length n...
(a) Randomly choose a vector $\mathbf{z}$ of length n and weight t
(b) Compute $\mathbf{c}' = \mathbf{m}\hat{G} \bigoplus \mathbf{z}$
Decryption: To decrypt ciphertext $\mathbf{c}'$ to message $\mathbf{m}$...
(a) Compute $\hat{\mathbf{c}} = \mathbf{c}' P^{-1}$
(b) Use the decoding algorithm D_G to decode $\hat{\mathbf{c}}$ to $\hat{\mathbf{m}}$
(c) Compute $\mathbf{m} = \hat{\mathbf{m}} S^{-1}$

- 256 KB public key for 2^{146} pre-quantum security
- 512 KB public key for 2^{187} pre-quantum security
- 1024 KB public key for 2^{263} pre-quantum security

Figure 1. Security analysis of McEliece public key sizes [9].

can be applied to $\hat{\mathbf{c}}$ to produce a k-length vector $\hat{\mathbf{m}}$, equal to $\mathbf{m}S$. The original message vector $\mathbf{m}$ is subsequently computed from the product of $\hat{\mathbf{m}}$ and S^{-1}, the inverse of the matrix S.

$$\mathbf{m} = \hat{\mathbf{m}} S^{-1}$$

The full key generation, encryption and decryption algorithms are summarized in Algorithm 1

2.3. Parameter sizes

McEliece originally proposed parameters of $n = 1024 = 2^{10}$, $t = 50$, and consequently a dimension of approximately $k = 1024 - 50 \times 10 = 524$ [11], corresponding to around 10^{149} Goppa polynomials, and an *'astronomical number of choices'* for S and P. These parameters have since been adjusted and result in a public key that can range from 100 kB upwards to over 1 MB. In her 2018 lecture at PQCRYPTO, Lange [9] summarized the security analysis of public key sizes shown in Figure 1.

3. Stern's attack

In 1989, Jacques Stern proposed a probabilistic method of finding codewords of low weight in linear codes [17]. Whilst not directly targeted towards breaking McEliece's cryptosystem, it was found that the ability to locate codewords of specific weight could be used to determine the error vector, of known weight, added during encryption. Stern's algorithm has been improved upon multiple times since its conception in 1989, specifically with the intention of attacking McEliece's cryptosystem. Notably, Canteaut and Chabaud [4], Canteaut and Sendrier [5], and Bernstein, Lange and Peters [2] all propose improvements on Stern's algorithm, or otherwise present their own attacks based on the principle of recovering minimum-weight codewords.

3.1. Stern's algorithm

The algorithm relies on the parameters p, $l \in \mathbb{N}$, which are chosen by the attacker. The choices of these parameters incur implications on both the complexity of the attack and probability of successful message recovery, which is discussed later in Subsection 2.3.

The algorithm is as follows:

Parameters:

- $n, k \in \mathbb{N}$, where $k \ll n$
- H, the $n \times k$ parity check Matrix
- C, the set of columns in H
- Columns $m_i \in C$
- Rows k_i of H. In the algorithm, each row k_i can be 'marked'; in doing so, the row is added to a list of visited indices (as in Step 1-e) and is used as a test condition throughout the algorithm
- $p, l \in \mathbb{N}$, algorithm parameters chosen by attacker.

Step 1: Perform Gaussian elimination on the matrix H as follows:
- for i: $= 1$ to $n-k$:
  (a) (a)Randomly choose, from a continuous uniform distribution, a column m_i of the matrix H, from the columns C that have not yet been chosen
  (b) (b)Select the first non-zero element in the column m_i, belonging to row k_i
  (c) (c)If row k_i is marked, repeat (b) for the remaining non-zero elements in m_i
  (d) (d)If all non-zero elements in m_i belong to a marked row, return to a and choose a new column
  (e) (e)When a non-zero element belonging to un-unmarked row k_i is chosen, mark the row k_i
  (f) (f)Transform the remaining elements in m_i to 0, excluding row k_i, through linear combination
  (g) (g)Add m_i to the set Z.

Step 2: Assign the indexes of the remaining columns of H to either of two sets X or Y with a probability of 1/2

Step 3: Randomly choose a set $J \subseteq \{x \in \mathbb{N} \mid x \leq n - k\}$ of l indices with values $\leq (n - k)$

Step 4: For all p-element subsets A of X (n.b. subsets A of X with p elements):
  (a) (a)Obtain the submatrix of H defined by $(H_{j,m})_{j\in J, m\in A}$ where the elements of J are the indexes of the rows and the elements of A are the indexes of the columns that make up the submatrix of H
  (b) (b)Compute the l-bit vector $\pi(A)$ by adding the columns of the submatrix obtained in (a)
  (c) (c)Perform the same operations for all p-element subsets B of Y.

Step 5: For all pairs $\pi(A) = \pi(B)$:

(1) Sum the columns of H given by the indexes in $A \cup B$ to produce a $n-k$ length vector V

(2) If V has a hamming weight of $w-2p$, return the vector x as follows:
  (a) (a)For all indexes m in x, set x_m to 1 if $m \in A \cup B$
  (b) (b)For all indexes k_i of V that equal 1, return the index m_i of the column z in H, where $z_{k_i} = 1$ and $z \in Z$. Set x_{m_i} to 1.

3.2. Applying Stern's algorithm to McEliece's cryptosystem

If binary vector $\mathbf{c}'$ (of length n) has a distance w from a codeword $\mathbf{c} \in C$, the weight of the vector $\mathbf{z} = \mathbf{c} \bigoplus \mathbf{c}'$ will also be w. Moreover, $\mathbf{z}$ is an element of $C \bigoplus \mathbf{c}'$, which can be regarded as the code that results from adding the vector $\mathbf{c}'$ to each element $\mathbf{c} \in C$.

An Eavesdropper, Eve, knows both the ciphertext $\mathbf{c}'$, and the number of errors t added during encryption. She therefore knows that $\mathbf{c}'$ has a distance t from its closest codeword $\mathbf{c} \in C$, and that the

code C has a minimum distance of $d = 2t + 1$, as it can correct up to t errors. There is therefore a unique codeword $\mathbf{c}$ with distance t from the ciphertext $\mathbf{c}'$. She also knows $\hat{G}$, the permuted generator matrix for code C. Because Eve knows the code $\hat{C}$ (generated by $\hat{G}$), she can then compute $\hat{C} \bigoplus \mathbf{c}'$ in which the only codeword with a weight t will be $\mathbf{z} = \mathbf{c} \bigoplus \mathbf{c}'$, since the weight of their sum is precisely their distance from each other. Therefore, knowing the weight of $\mathbf{z} \in \hat{C}$, Eve can apply Stern's algorithm to recover it probabilistically and, from this, derive the codeword $\mathbf{c} \in C$ used in encryption. This is not quite the full plaintext recovery, but decoding the codeword $\mathbf{c} \in C$ to its corresponding message vector $\mathbf{m}$ is a trivial matter, which is discussed later.

3.3. Probability and complexity

The probability of recovering successful parameters during Stern's algorithm is given as the product of the following three equations:

$$\binom{w}{2p}\binom{n-w}{k-2p}/\binom{n}{k} \tag{2}$$

$$\binom{2p}{p}/4^p \tag{3}$$

$$\binom{n-k-w+2p}{l}/\binom{n-k}{l} \tag{4}$$

With the parameters presented by Stern ($n = 300$, $k = 150$, $w = 20$, $p = 3$, $l = 12$) the probability of successfully finding a codeword of weight w is 0.0031. However, replacing these with McEliece's parameters ($n = 1024$, $k = 524$, $w = 50$) and retaining the values $p = 3$, $l = 12$, this becomes 2×10^{-10}. The probability increases with larger values of p and smaller values of l but, additionally, the complexity also increases.

Stern acknowledges the steps with the highest complexity to be step 1 (Equation 5), step 4 (Equation 8) and step 5 (Equation 7).

$$1/2(n-k)^3 + k(n-k)^2 \tag{5}$$

$$2lp\binom{k/2}{p} \tag{6}$$

$$2p(n-k)\binom{k/2}{p}^2/2^l \tag{7}$$

With Stern's parameters ($n = 300$, $k = 150$, $w = 20$, $p = 3$, $l = 12$), the number of operations become 5×10^6 for step 1, 5×10^5 for step 4 and 10^9 for step 5. With McEliece's parameters, however, this becomes 2×10^8 for both steps 1 and 4 and 6×10^{12} for step 5. Again, decreasing p and increasing l reduces the complexity of these steps, but also makes a successful attack less probable, therefore introducing a substantial tradeoff between complexity and probability.

4. Lee and Brickell's Attack

One year before Stern, in 1988, Lee and Brickell also proposed a probabilistic attack on McEliece's cryptosystem [10]. This attack built upon McEliece's suggestion of randomly choosing k bits from the ciphertext in hope that none would be in error.

An attacker can randomly select k bits from the n-length ciphertext $\mathbf{c}'$ to form the k-length vector $\mathbf{c}_k$. They can also recover the $k \times k$ submatrix $\hat{G}_k$, where the columns of $\hat{G}_k$ correspond to the k columns of the generator matrix $\hat{G}$. If there is no error in the k bits chosen to make up the vector $\mathbf{c}_k$, the original message $\mathbf{m}$ can be recovered from $\mathbf{m} = \mathbf{c}_k\hat{G}_k^{-1}$.

The codeword $\mathbf{c} \in \hat{C}$ corresponding to $\mathbf{m}$ can be found from $\mathbf{c} = (\mathbf{c}_k\hat{G}_k^{-1})\hat{G}$, which is equivalent to $\mathbf{c} = \mathbf{m}\hat{G}$ if and only if $\mathbf{m} = \mathbf{c}_k\hat{G}_k^{-1}$. As discussed in Section 3.2, if a vector $\mathbf{c}'$ has a hamming distance t from its closest codeword $\mathbf{c} \in C$, the weight of the vector $\mathbf{z} = \mathbf{c} \bigoplus \mathbf{c}'$ will also be t. Therefore, if $\mathbf{m} = \mathbf{c}_k\hat{G}_k^{-1}$, the vector $\mathbf{z}$ of weight t can be found from $\mathbf{z} = \mathbf{c}' \bigoplus (\mathbf{c}_k\hat{G}_k^{-1})\hat{G}$. This principle can be used to systematically check the choice of $\mathbf{c}_k$ as, if $\mathbf{m} \neq \mathbf{c}_k\hat{G}_k^{-1}$, the hamming weight of $\mathbf{z} = \mathbf{c}' \bigoplus (\mathbf{c}_k\hat{G}_k^{-1})\hat{G}$ will be greater than t. A new choice of k can be made repeatedly until the weight of $\mathbf{z}$ becomes t.

Lee and Brickell further generalize this by introducing a k-length error vector $\mathbf{e}_k$, which is added to $\mathbf{c}_k$ during the attack.

4.1. Lee and Brickell's Algorithm

The algorithm relies on the parameter j, which is chosen by the attacker. Once again, the value of this parameter affects the complexity of the attack and the probability of successful message recovery.

The algorithm is as follows:

Parameters:

- $n, k \in \mathbb{N}$,where $k \ll n$
- $j \in N$, algorithm parameter chosen by the attacker
- $\mathbf{c}'$, the n-bit ciphertext
- $\hat{G}$, the permuted generator matrix, as in Algorithm 1.

Step 1: Randomly select k bits from the n-bit ciphertext $\mathbf{c}'$ to form the k-bit vector $\mathbf{c}_k$. Set $\hat{G}_k$ to the submatrix of $\hat{G}$, where each column in $\hat{G}_k$ corresponds to the k columns in $\hat{G}$.

Step 2: For all k-bit error vectors $\mathbf{e}_k$ with weight less than or equal to j:

(1) • If the weight of $(\mathbf{c} + \mathbf{c}_k\hat{G}_k^{-1}G) + \mathbf{e}_k(\hat{G}_k^{-1}\hat{G})$ is less than or equal to t:
$\mathbf{m} = (\mathbf{c}_k + \mathbf{e}_k)\hat{G}_k^{-1}$.

(2) *Step 3:* If all k-bit error vectors $\mathbf{e}_k$ have been used, return to *step 1.*

4.2. Probability and complexity

Lee and Brickell state that the probability of the chosen k-bit vector $\mathbf{c}_k$ containing i errors is given by

$$Q_i = \binom{t}{i}\binom{n-t}{k-i}/\binom{n}{k} \tag{8}$$

And that therefore the probability of a successful attack is

$$\sum_{i=0}^{j} Q_i \tag{9}$$

Additionally the number of executions in step 2 is equal to the number of k-bit error vectors with weight less than or equal to j

$$N_j \sum_{i=0}^{j} \binom{k}{i} \tag{10}$$

Increasing the value of the parameter j therefore results in a greater likelihood of a successful attack, but also increases the complexity of the algorithm by introducing a greater number of k-length error vectors to be iterated through in step 2. With McEliece's parameters ($n = 1024$, $k = 524$, $t = 50$), and a small value of $j = 3$, the probability of a successful attack (Equation 9) is 1×10^{-13} and the complexity of step 2 (Equation 10) is 2×10^7.

5. A novel attack

This section presents a new, novel attack on McEliece's cryptosystem, adapting and building upon the two attacks reviewed above. Characteristics shared by both Stern's, and Lee and Brickell's attacks are exploited to create a new algorithm. This attack is subsequently analysed and suggestions for further improvement presented.

5.1. Overview

The probabilistic nature shared by Stern's algorithm and Lee and Brickell's attack can be exploited to combine the two attacks into one. Specifically, during Gaussian elimination in step 1 of Stern's algorithm (see Section 3.1), $n-k$ of the n columns in the parity check matrix H are chosen randomly, leaving a remaining k columns unselected ($n-(n-k)=k$). In step 1 of Lee and Brickell's attack (see Section 4.1), k bits of the n-length ciphertext $\mathbf{c}'$ are chosen, also at random. Therefore, the first step of both algorithms is practically identical.

The Gaussian elimination performed in Stern's algorithm randomly selects $n-k$ of n columns in H and therefore inadvertently chooses k of n columns, also randomly, by not selecting them. Therefore, after randomly selecting $n-k$ columns, the indexes of the remaining k columns can then be used to obtain the k-length vector $\mathbf{c}_k$ in step 1 of Lee and Brickell's attack. After completing the Gaussian elimination in step 1 of Stern's attack, Lee and Brickell's attack can consequently be performed by using the k bits obtained as discussed above.

As the k indexes used in Lee and Brickell's attack are dependent on the output of Stern's first step, an additional adjustment is made. Previous selections of the k bits are stored in a cache and if the k columns chosen by Gaussian elimination have already been used previously, Lee and Brickell's attack will not run for that iteration. This is to ensure that Lee and Brickell's attack does not run unnecessarily if a particular selection of k bits has already been examined.

A further addition can be seen in Step 2(c). In Lee and Brickell's attack, the message vector $\mathbf{m}$ is recovered by ($\mathbf{m} = (\mathbf{c}_k + \mathbf{e}_k)\hat{G}_k^{-1}$). However, in Step 1 of the adapted attack, the ciphertext $\mathbf{c}'$ is appended to the rows of the generator matrix $\hat{G}$, resulting in a $(k+1) \times n$ generator matrix, and so the original means of recovering $\mathbf{m}$ becomes invalid. Whilst the operations could be performed on the original generator matrix, the output of Step 2(c) (vector $\mathbf{z}$ of weight t) is the error vector added during encryption. Stern's algorithm also outputs the same error vector and proceeds to decode $\mathbf{c} = \mathbf{c}' \bigoplus \mathbf{z}$ to $\mathbf{m}$. Therefore, after successfully recovering the codeword $\mathbf{z}$ by Lee and Brickell's attack, the same decoder is used as in Stern's. This saves the operation $(\mathbf{c}_k + \mathbf{e}_k)\hat{G}_k^{-1}$ from having to be performed.

5.2. The algorithm

The new attack can be generalized as follows:

Parameters:

- $n, k \in \mathbb{N}$,where $k \ll n$
- $\mathbf{c}'$, the n-bit ciphertext
- $\hat{G}$, the permuted generator matrix as in Algorithm 1
- $p, l, j \in \mathbb{N}$, algorithm parameters chosen by attacker
- K, a set initialized to $\emptyset$.

Step 1: (a) (a)Append the n-length ciphertext $\mathbf{c}'$ to the rows of the generator matrix $\hat{G}$ and compute the parity check matrix H corresponding to $\hat{G}$.
(b) (b)Perform Stern's Gaussian elimination on the matrix H (Subsection 3.1: step 1).
(c) (c)Add the indexes of the k columns not selected by Gaussian elimination to the set K.

Step 2: If set K has not been selected previously…

(a) (a)Obtain the k-bit vector $\mathbf{c}_k$ by selecting k bits of the ciphertext $\mathbf{c}'$ corresponding to the elements in set K

(b) (b)Set $\hat{G}_k$ to the submatrix of $\hat{G}$, where each column in $\hat{G}_k$ corresponds to the k columns in $\hat{G}$

(c) (c)For all k-bit error vectors $\mathbf{e}_k$ with weight less than or equal to j:

  (a) If the weight of n-length vector $\mathbf{z} = (\mathbf{c}' + \mathbf{c}_k\hat{G}_k^{-1}G) + \mathbf{e}_k(\hat{G}_k^{-1}\hat{G})$ is less than or equal to t:

  (b) $\mathbf{z}$ is the error-vector added during encryption, therefore $\mathbf{c} = \mathbf{c}' \bigoplus \mathbf{z}$

  (c) Jump to *Step 4*.

(d) (d)If no $\mathbf{m}$ found, add K to a cache of previously selected values for K and proceed to *Step 3*. The next random selection of $\mathbf{c}_k$ is performed at the next iteration if Stern's attack fails.

Step 3: Perform the remaining steps of Stern's algorithm Section (3.1: steps 2 – 5)

(a) (a)If Stern's algorithm outputs a vector $\mathbf{z}$ of weight t:

  (a) $\mathbf{z}$ is the error-vector added during encryption, therefore $\mathbf{c} = \mathbf{c}' \bigoplus \mathbf{z}$

  (b) Jump to *Step 4*

(b) (b)If no output found, return to *Step 1*.

Step 4: Decode $\mathbf{c}$ to $\mathbf{m}$ according to the code $\hat{C}$ defined by $\hat{G}$.

5.3. Probability and complexity

As discussed above, randomly selecting $n-k$ columns of the parity check matrix is precisely the selection of k out of n indexes, and we can take these k indexes as our random selection for the Lee–Brickell algorithm. We can therefore talk about the success of the two algorithms on the same trial. Witness cases suggest that there are some cases in which both succeed, so the probability of success of the novel algorithm does not necessarily attain the theoretical maximum of the sum of the probabilities of the two original algorithms. We estimate the probability experimentally.

Assuming each trial is independently distributed, and that the probability p of success is constant on each trial, one can show that the expected number of trials until success is $1/p$. To estimate this success probability, we average the number of trials until success on several iterations of the algorithm, calculating a mean N. Assuming this gives us a good estimate for the true expected number of trials until success, we estimate the probability of success of an individual trial as $1/N$. We can compare this to the theoretical success probabilities of each algorithm, as well as their sum, which is the theoretical maximum success probability of our algorithm.

The best-case probability of Stern's attack is detailed in Section 3.3 as the product of Equations (2), (3) and (4). Similarly, the probability of success for Lee and Brickell's attack is detailed in Section 4.2 and given by Equation (9).

A theoretical maximum for the probability of successful attack when combining the two algorithms is therefore the sum of these two probabilities (Equation 2×Equation 3×Equation 4+ Equation 9)

$$\frac{\binom{w}{2p}\binom{n-w}{k-2p}}{\binom{n}{k}} \times \frac{\binom{2p}{p}}{4^p} \times \frac{\binom{n-k-w+2p}{l}}{\binom{n-k}{l}} + \sum_{i=0}^{j} Q_i \tag{11}$$

However, combining these two algorithms also incurs a penalty with respect to complexity, outlined for both Stern's algorithm and Lee and Brickell's attack in Sections 3.3 and 4.2. Whilst this is a drawback, the parameters of p, l and j can be adjusted to increase or decrease the complexity of either attack, making the system overall rather flexible and adjustable.

Table 1. Expected analysis: $n = 254$, $k = 94$, $t = 20$, $p = 2, l = 12, j = 2$.

	Stern	Lee & Brickell	Combined
Probability	0.0052	0.0059	0.0111
Iterations	190.69	170.11	89.91

Table 2. Expected analysis: $n = 1024$, $k = 524$, $t = 50$, $p = 2$, $l = 12, j = 2$.

	Stern	Lee & Brickell	Combined
Probability	3.45×10^{-12}	1.24×10^{-13}	3.57×10^{-12}
Iterations	2.90×10^{11}	8.08×10^{12}	2.70×10^{11}

Table 3. Expected analysis: $n = 1024, k = 524, t = 50, p = 4$, $l = 12, j = 4$.

	Stern	Lee & Brickell	Combined
Probability	1.10×10^{-8}	3.20×10^{-11}	1.09×10^{-8}
Iterations	9.21×10^{7}	3.12×10^{10}	9.18×10^{7}

With Goppa code parameters $n = 254$, $t = 20$, $k = 94$ and algorithm parameters $p = 2$, $l = 12$, $j = 2$ (used later in analysis), the best-case probability and expected number of iterations to a successful attack is given in Table 1. A similar analysis is shown for McEliece's full parameters in Table 2, which maintains the values for $p = 2$, $l = 12$, $j = 2$. With these parameters, the complexity for step 5 of Stern's algorithm (Equation 7) is 5×10^8. Increasing the algorithm parameters to $p = 4$, $p = 12$, $j = 4$ greatly increases the probability of successful attack, shown in Table 3. However, this also raises the complexity of Equation (7) to 3×10^{16}. The implications of this are discussed further in Section 5.5.

This is, however, just a fundamental speculation of the probability and complexity incurred by combining the two attacks and further study should be conducted into analysing the method more comprehensively.

5.4. Analysis

Three separate analyses were conducted on each algorithm with the aim of experimentally analysing both their probabilities for successful message recovery, as well as their complexities. Due to constraints in both time and resources, attacks could not be performed on McEliece's cryptosystem using the full parameters proposed ($n = 1024, t = 50, k = 524$). Instead, smaller parameters were used that would allow for an attack to be performed in a feasible amount of time, whilst aiming to remain large enough for a comparison to still be made.

5.4.1. Analysis 1

In accordance with the argument proposed in Section 5.3, each algorithm was run 50 times and an average taken for the time and number of iterations performed for a successful attack. After each run, a new public key, private key and message vector were randomly generated to avoid any potential bias that may arise from specific configurations of keys or messages.

The parameters chosen were $n = 254$, $t = 20$, $k = 94$, and the variable algorithm parameters p, l, j remained constant within each attack. The chosen values for Stern's variables were $p = 2$, $l = 12$, and the value chosen for Lee and Brickell's variable was $j = 2$. The values of p, l, j were chosen to reduce the complexity of each algorithm.

Table 4. Analysis: $n = 254$, $k = 94$, $t = 20$, $p = 2$, $l = 12, j = 2$.

	Stern	Lee & Brickell	Novel
Av. Iterations	194.37	149.25	71.42
Av. Time	1467.10	1098.76	1149.93

Table 5. Stern.

j \ p	2	3	4	5 (Theory)
2	0.0146	0.0021	0.0001	0.00001
3	0.0146	0.0021	0.0001	0.00001
4	0.0146	0.0021	0.0001	0.00001
5	0.0146	0.0021	0.0001	0.00001

Table 6. Lee & Brickell.

j \ p	2	3	4	5
2	0.484	0.484	0.484	0.484
3	0.833	0.833	0.833	0.833
4	0.909	0.909	0.909	0.909
5	0.937	0.937	0.937	0.937

Table 7. Novel.

j \ p	2	3	4	5
2	0.400	0.556	0.390	0.484
3	0.667	0.698	0.732	0.750
4	0.833	0.937	0.833	0.937
5	0.968	0.968	0.937	0.967

5.4.1.1. Remarks. The results produced in Table 4 align with the expected values (Table 1) to perform a successful attack on McEliece's cryptosystem with parameters $n = 254, k = 94, t = 20, p = 2$, $l = 12, j = 2$.

Additionally, the results of 50 runs suggest that the novel attack is able to successfully recover a message faster than Stern's algorithm, although this speculation is limited to the specific parameters chosen for n, k, t. This analysis does, however, suggest that the Novel attack is slower than Lee and Brickell's attack but, in spite of this, the number of iterations to successful attack are still greatly reduced from Lee and Brickell's

5.4.2. Analysis 2

Each algorithm was run 30 times for varying values of p and j, ranging from 2 to 5, whilst the value of l remained constant at $l = 12$. The chosen code parameters were $n = 128, k = 23, t = 15$. Although it is acknowledged that these parameters are small, and not necessarily representative of the full values proposed by McEliece, the method provides a rudimentary insight to the performance of each algorithm across varying values of p and j.

The results for Stern's algorithm are shown in Table 5. It should be noted that the results for $p = 5$ are calculated theoretically (see Section 3.3). The results for Lee and Brickell's algorithm are shown in Table 6, and the results for the Novel algorithm are shown in Table 7. The results of Stern's, and Lee and Brickell's probability analysis were added together to produce Table 8, which represents the theoretical maximum for the Novel attack.

5.4.2.1. Remarks. With the reduced parameters of n, k, t, it appears that Lee and Brickell's algorithm dominates Stern's, as evidenced by comparing Tables 5 and 6. Nonetheless, the results shown in the

Table 8. p(Stern)+p(Lee & Brickell).

j \ p	2	3	4	5
2	0.498	0.486	0.484	0.484
3	0.848	0.835	0.833	0.833
4	0.924	0.911	0.909	0.909
5	0.952	0.885	0.883	0.883

Table 9. Analysis 3: $n = 512$, $k = 332$, $t = 20$, $p = 2$, $l = 12$, $j = 2$.

	Stern	Lee & Brickell	Novel
Iterations	361	221	139
1 / Iterations	2.77×10^{-3}	4.52×10^{-3}	7.19×10^{-3}

Table 10. Analysis 3: $n = 1024$, $k = 524$, $t = 50$, $p = 2, l = 12, j = 2$.

	Stern	Lee & Brickell	Novel
Iterations	22	52	16
1 / Iterations	0.045	0.019	0.063

analysis of the Novel attack (Table 7) are very close to the expected values, represented by Table 8, with a greatest difference of 0.181 at $p = 2, j = 3$. It is therefore suggested that the Novel attack comes close to combining the two algorithms' probabilities; however, this should be experimented further on a larger set of values for n, k, t and a greater range of values of p, l, j for any concrete conclusion to be drawn.

5.4.3. Analysis 3

Whilst unable to successfully recover a message when attacking larger parameters, each algorithm was still able to run, albeit not to completion. Therefore, to gain an insight to the algorithms' performance against larger parameters, each attack was run for exactly 24 hours and the number of iterations N performed within this period was recorded.

We therefore naively estimate that the expected number of iterations E until success is greater than N.

$$E > N \tag{12}$$

As discussed in Section 5.3, under certain assumptions the probability for successful message recovery p is equal to the reciprocal of the number of trials until success. For the expected number of trials, the probability is expressed as

$$p = 1/E \tag{13}$$

Therefore, using Equations (12) and (13), we can derive an estimate for an upper bound on the probability as

$$p < 1/N \tag{14}$$

Each attack was run for 24 hours with fixed algorithm parameters of $p = 2, l = 12, j = 2$. This was performed firstly on parameters $n = 512, k = 332, t = 20$ and then on McEliece's proposed parameters of $n = 1024, k = 524, t = 50$. The results of these are shown in Tables 9 and 10 respectively, which display both the number of iterations N and the reciprocal $1/N$.

5.5. Further improvements

The relationships between the parameters p, l, j, and the probability and complexity of the attack is discussed in Section 5.3 and also earlier in Sections 3.3 and 4.2. It is also suggested by the second analysis (Section 5.4.2) that these perform differently on different code sizes; where, in Table 5, it can be seen that lower values of p produce a higher probability when attacking the smaller parameters $n = 128$, $k = 23$, $t = 15$. It is clear that each of these algorithm parameters has a great bearing on probability and complexity and, therefore, the attack would benefit from optimizing these parameters prior to beginning the algorithm. Further study could be made into the relationship between these parameters and their optimal values. From this, a preamble to the attack could be created, in which these parameters are adaptively optimized according to resources available and the size of the code being attacked.

Additionally, whilst step 2 of the algorithm is dependent on the output of step 1, the rest of the algorithm (steps 3–4) do not depend on step 2. Therefore, after step 1 is complete, step 2 can run parallel to the rest of the algorithm until completion, resulting in a substantial speed up.

Extensive study has already been conducted on both Stern's algorithm and Lee and Brickell's attack and the novel attack proposed in this report would benefit from incorporating improvements that have already been made in prior study. In particular, Bernstein, Lange and Peters made improvements to Stern's algorithm resulting in a successful attack being performed in $2^{60.55}$ bit operations [2], eclipsing improvements made previously by Canteaut and Sendrier [5] ($2^{64.1}$ bit operations). On a normal computer, a successful attack is cited to take 7,400,000 days, but this is reduced to 2 months on a larger computer cluster. These improvements to Stern's algorithm can be incorporated into the attack proposed here and would greatly reduce the complexity and increase the probability of successful message recovery.

5.6. Concluding remarks

This report proposes a novel attack on McEliece's cryptosystem that combines those proposed by Stern, and Lee and Brickell. A basic analysis suggests that the probability of successful attack compounds those of Stern's, and Lee and Brickell's attacks. Additionally, the results of Section 5.4.1 suggest that the Novel attack may be able to successfully recover a message faster than Stern's algorithm, although slower than Lee and Brickell's. Whilst the analysis conducted provides no concrete evidence that the Novel attack has any advantage over the two original algorithms, it was most appropriate for the scope of the research, accommodating for limitations in both time and resources. It should also be noted that McEliece's cryptosystem still remains unbroken, and the inability to conduct a thorough analysis on significant parameters is a supporting argument for its continued security.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

[1] E.R. Berlekamp, *Goppa codes*, IEEE Transactions on Information Theory 19 (5) (1973), pp. 590–592.
[2] D.J. Bernstein and T. Lange and C. Peters, *Attacking and Defending the McEliece Cryptosystem*, uchmann J., Ding J, eds., Post-Quantum Cryptography. PQCrypto 2008. Lecture Notes in Computer Science, Vol. 5299, 2008.
[3] A. Canteaut and H. Chabanne, *A further improvement of the work factor in an attempt at breaking McEliece's cryptosystem* RR-2227, INRIA, 1994.
[4] A. Canteaut and F. Chabaud, *A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511*. IEEE Transactions on Information Theory, Vol. 44, no. 1, 1998, pp. 367–378.
[5] A. Canteaut and N. Sendrier, *Cryptanalysis of the Original McEliece Cryptosystem*, Ohta K., Pei D, eds., Advances in Cryptology – ASIACRYPT 1998. Lecture Notes in Computer Science, Vol. 1514, 2000.

[6] W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, IT-22, 1976, pp. 644–654.
[7] V.D. Goppa, *A new class of linear error-correcting codes*, Problemy Peredachi Informatsii 6 (1971), pp. 41–49.
[8] L.K. Grover, *A fast quantum mechanical algorithm for database search*. STOC, 1996, pp. 212–219.
[9] T. Lange, *Code-Based Cryptography*, PQCRYPTO, 2018, http://hyperelliptic.org/tanja/vortraege/20180628.pdf.
[10] P.J. Lee and E.F. Brickell, *An Observation on the Security of McEliece's Public-Key Cryptosystem*, Barstow D. et al. eds., Advances in Cryptology–EUROCRYPT 1988. Lecture Notes in Computer Science, Vol. 330, 1988, pp. 275–280.
[11] R.J. McEliece, *A Public-Key Cryptosystem Based on Algrebraic Coding Theory*. Deep Space Network Progress Report, 1978, pp. 114–116.
[12] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
[13] National Institute of Standards and Technology, L. Chen, S. Jordan, Y-K. Liu, D. Moody, R. Peralta, R. Perlner, D. Smith-Tone, eds., *Report on Post-Quantum Cryptography*. NISTIR 8105, 2016.
[14] National Institute of Standards and Technology, G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, D. Smith-Tone, eds., *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process*. NISTIR 8309, 2020.
[15] R.L. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Commun. ACM 21 (2) (1978), pp. 120–126.
[16] P.W. Shor, *Polynomial-Time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput. 26 (5) (1997), pp. 1484–1509.
[17] J. Stern, *A Method for Finding Codewords of Small Weight*, in *Coding Theory and Applications. Coding Theory 1988*, G. Cohen, J. Wolfmann, eds., Lecture Notes in Computer Science, Vol. 388, 1989, pp. 106–113.