A Study of the McEliece PKE

Shaan Vaidya Srijit Dutta

Spring 2018

1 Introduction

This report is part of a reading project for CS 406: Cryptography and Network Security. Robert McEliece proposed a public key cryptosystem [3] in 1978. It was the first such PKE to use randomness in the encryption algorithm. The McEliece PKE is based on error-correcting linear codes. The original construction uses binary Goppa codes [1] but the algorithm can be used with any 'good enough' linear codes. This PKE is still considered secure with the right parameters. Even though the algorithm has rarely been used in practice due to large key sizes, it is interesting to study because it is being considered one of the best candidates for a post quantum secure PKE (i.e. when quantum computers take over). First, we present basic notions of coding theory; then move on to Goppa codes (which were used in the original construction); an overview of the algorithm and finally a high level discussion of the security of the PKE.

2 Coding Theory

Coding theory is used to ensure accurate transmission of information across a possibly noisy channel from the sender to the receiver. The purpose is to efficiently attach some additional information to the message so that the receiver can correct as many errors as possible. The encoder encodes the message to a codeword by adding some redundancy and the decoder decodes the codeword to correct errors which the noisy channel may have introduced and obtain the message from the encoder.

2.1 Linear Codes

A linear code is an error correcting code in which any linear combination of codewords is also a codeword. Linear codes permit more efficient encoding and decoding algorithms to be used.

Definition 1. *Let $\mathcal{F}$ be a finite field. An [n,k] linear code $\mathcal{C}$ is a k-dimensional subspace of $\mathcal{F}^n$, that is, for every two codewords, c_1 and $c_2 \in \mathcal{C}$ and scalars a_1,$a_2 \in \mathcal{F}$ the codeword $a_1c_1 + a_2c_2 \in \mathcal{C}$.*

Definition 2. *The minimum distance of $\mathcal{C}$ is the minimum distance between any two distinct codewords in $\mathcal{C}$*

$$d = \min_{c_1 \neq c_2} d(c_1, c_2) \tag{1}$$

If $\mathcal{C}$ is a $[n, k]$ linear code with minimum distance d, we say that $\mathcal{C}$ is a $[n, k, d]$ linear code.

Definition 3. *Generator Matrix : For a [n,k] linear code $\mathcal{C}$ over $\mathcal{F}$, its generator matrix G is a $k \times n$ matrix over $\mathcal{F}$ whose rows form the basis of $\mathcal{C}$.*

The generator matrix for a given linear code need not be unique.

Definition 4. *Parity Check Matrix : Let $\mathcal{C}$ be a [n,k] linear code over field $\mathcal{F}$. A parity check matrix of $\mathcal{C}$ is an (n-k) $\times$ n matrix H over $\mathcal{F}$ such that for every $c \in \mathcal{F}^n$,*

$$c \in \mathcal{C} \iff Hc^T = 0 \tag{2}$$

From the above definition of parity matrix it can be seen that if G and H are the generator and parity check matrices of the same linear code $\mathcal{C}$, then $HG^T = GH^T = 0$.

Definition 5. *Error Correcting Code : Let $\mathcal{C}$ be an [n,k,d] linear code over $\mathcal{F}$ with generator matrix G. We say that $\mathcal{C}$ can correct up to t errors, if there exists a decoding algorithm $\mathcal{D} : \mathcal{F}^n \to \mathcal{C}$ such that for every $u \in \mathcal{F}^n$ and every vector $e \in \mathcal{F}^n$ with weight $w(e) \leq t$, the word*

$$y = uG + e \tag{3}$$

is always correctly decoded as $\mathcal{D}(y) = u$.

A common way to implement the above is by using the nearest codeword decoding method, assuming there is a unique closest codeword.

2.2 Syndrome Decoding

Syndrome Decoding is an efficient method for decoding binary linear codes. This is similar to the minimum distance method but the look-up table is much reduced in size. However, it is an NP-complete problem if the number of errors is not bounded. Suppose a codeword $c \in \mathcal{F}^n$ is sent over the channel, then the decoder receives a noisy message y, where $y = c+e$. The decoder wants to find out the message c. Given y and the parity check matrix H, we compute the syndrome

$$s = Hy^T = H(c + e)^T = He^T \tag{4}$$

The syndrome is looked up in the table and the corresponding error vector e is obtained. Thus we can now find the codeword as $c = y - e$. The syndromes of the error vectors are computed and stored in a table, along with the vectors, before the process. Syndrome decoding is efficient in the sense that time is saved over making a full table of all possible words and error vectors, which requires 2^n entries to be computed compared to the symbol table which has 2^{n-k+1} entries to be computed. However, this bound is still very large and is in fact the problem on which the McEliece cryptosystem relies. Only by knowing the actual code can one find out the actual codeword by a fast decoding algorithm.

3 Goppa Codes

Definition 6. *Binary Goppa Code : Let n, m and t be positive integers and let*

$$g(X) = \sum_{i=0}^{t} g_i X^i \in \mathcal{F}_{2^m}[X] \tag{5}$$

be a monic polynomial of degree t. Let $\mathcal{L} = (\alpha_1, \ldots, \alpha_n) \in \mathcal{F}_{2^m}^n$ *be a tuple of n distinct elements form* $\mathcal{F}_{2^m}^n$*, such that*

$$g(\alpha_i) \neq 0, \forall i : 1 \leq i \leq n. \tag{6}$$

The Goppa code $\Gamma(\mathcal{L}, g)$ *consists of all elements* $c = (c_1, \ldots, c_n) \in \{0,1\}^n$ *that satisfy*

$$\sum_{i=1}^{n} \frac{c_i}{X - \alpha_i} \equiv 0 \, mod \, g(X) \tag{7}$$

The above equation can also be written as

$$\sum_{i=1}^{n} c_i \{(X - \alpha_i)^{-1}\}_g = 0, \tag{8}$$

where $\{(X - \alpha_i)^{-1}\}_g$ is the inverse of $(X - \alpha_i)$ in the algebra of polynomials *mod* $g(X)$ and can be calculated as

$$\{(X - \alpha_i)^{-1}\}_g = \frac{g(X) - g(\alpha_i)}{X - \alpha_i}(g(\alpha_i)^{-1}) \tag{9}$$

Due to this, we can also define binary Goppa codes using a parity check matrix as follows:

Definition 7. *The Goppa code* $\Gamma(\mathcal{L}, g)$ *consists of all vectors* $\boldsymbol{c} = (c_1, \ldots, c_n) \in \{0,1\}^n$ *such that*

$$H\boldsymbol{c}^t = 0 \tag{10}$$

where

$$H = \begin{bmatrix} \frac{1}{g(\alpha_1)} & \cdots & \frac{1}{g(\alpha_n)} \\ \frac{\alpha_1}{g(\alpha_1)} & \cdots & \frac{\alpha_n}{g(\alpha_n)} \\ \cdots & \ddots & \cdots \\ \frac{\alpha_1^{t-1}}{g(\alpha_1)} & \cdots & \frac{\alpha_n^{t-1}}{g(\alpha_n)} \end{bmatrix} \tag{11}$$

Note that $\frac{1}{g(\alpha_i)}$ *is the inverse element in the field* $\mathcal{F}_{2^m}$

If the polynomial $g(X)$ has no factors (irreducible), all $\alpha \in \mathcal{F}_{2^m}$ satisfy $g(\alpha) \neq 0$. The corresponding code is called an irreducible binary Goppa code. Goppa codes are the primary choice of the McEliece cryptosystem for many reasons. They have a fast polynomial time decoding algorithm. Also any irreducible polynomial can be used to create a Goppa code, but the generator matrices are almost random. Thus, though easily generated, they are difficult to find. As the length of the code and degree of the generating polynomial increases, the number of Goppa codes increase exponentially.

Finding an irreducible polynomial for setting up Goppa codes is not much difficult, as there are about $\frac{2^m t}{t}$ polynomials of degree t over $\mathcal{F}(2^m)$, thus a randomly chosen polynomial will be irreducible with probability $1/t$. Also there exists fast algorithms for testing irreducibility.

3.1 Fast Decoding (Patterson's Algorithm)

Below is a brief outline of the fast decoding algorithm [4] used to find the error vector in McEliece :

- Given received codeword y find syndrome $s(x) = \sum_{i=1}^{n} \frac{y_i}{x-\alpha_i}$ mod g(x)
- Find $h(x)$, inverse of $s(x)$ in the algebra of polynomials mod $g(x)$
- Find $d(x)$ such that $d^2(x) \equiv h(x) + x \; mod \; g(x)$
- Find $a(x), b(x)$ of least degree such that

$$d(x)b(x) \equiv a(x) \; mod \; g(x) \; \textit{(Extended Euclidean Algorithm)}$$

- $\sigma(x) = a^2(x) + xb^2(x)$
- $\sigma(x)$ is used to determine the set of error locations $E = \{i \mid \sigma(\alpha_i) = 0\}$
- Error positions are i such that $i \in E$
- Codeword c is found as $y - e$

4 The McEliece PKE

The McEliece PKE uses error-correcting linear codes for encrypting messages. The private key contains the description of the linear code, chosen at the time of key generation, and the public key is obtained by "scrambling" the code i.e. by making it hard to distinguish from a completely random linear code. Decryption requires that the existence of a fast decoding algorithm for the chosen linear code. Security comes from the fact that knowing the underlying linear code (private key) helps in fast decryption, but it is hard to decrypt without that knowledge.

4.1 Construction

Let $\mathbf{C}$ be a $[n, k]$-linear code with a fast decoding algorithm $\mathcal{D}$ that can correct t or fewer errors (in the original construction: a Goppa code).
Private key: a $k \times k$ generator matrix $\mathbf{G}$ for $\mathbf{C}$, a $k \times k$ invertible matrix $\mathbf{S}$ (the scrambler) and a $n \times n$ permutation matrix $\mathbf{P}$ (and the decoding algorithm $\mathcal{D}$
Public key: $(\mathbf{G}^{'} = \mathbf{SGP} \; , t)$.
Encryption : If Alice wants to send a message to Bob, she first needs to break it down into k-bit blocks. If $\mathbf{u}$ is one such block, she first creates $\mathbf{z}$ — a random binary n-bit vector of weight t (randomly placed t 1's) where t is the degree of the generating polynomial of the Goppa code or the lower bound of the error correcting ability of the code in the general case. Then she sends $\mathbf{x} = \mathbf{uG}^{'} + \mathbf{z}$ to Bob.
Decryption : On receiving $\mathbf{x}$, Bob first computes

$$\mathbf{xP}^{-1} = (\mathbf{uG}^{'} + \mathbf{z})\mathbf{P}^{-1} = \mathbf{uSG} + \mathbf{zP}^{-1} = \mathbf{uSG} + \mathbf{z}^{'}$$

where $\mathbf{z}^{'}$ is also a vector of weight t. Now, $\mathbf{xP}^{-1}$ is a codeword in the Goppa code chosen. Bob can now use the fast decoding algorithm $\mathcal{D}$ to get $\mathbf{uSG}$. Multiply $\mathbf{G}^{-1}$ to get $\mathbf{uS}$ and now, since $\mathbf{S}$ is invertible, Bob can get back $\mathbf{u}$.

4.2 Drawbacks

- McEliece suggested the values $n = 1024 = 2^{10}$ and $k = 524 \simeq 2^9$ for the Goppa code parameters and therefore the size of the public key would be about $n \times k \simeq 2^{19}$ which is very large from the point of view of implementation
- The size of the ciphertext is quite large compared to that of the plaintext and thereby reduces the transmission efficiency and opens the door for transmission errors.
- Decryption algorithm cannot be used as an encryption to get signature schemes (unlike RSA) as the algorithm would not produce an output for inputs which have more than t error bits (with respect to any codeword) and thus produces an output for a very small group of inputs.

5 Security

The security of the system rests on how difficult it is, for an adversary who knows $\mathbf{G}^{'}$ and intercepts $\mathbf{x}$, to recover $\mathbf{u}$. The security of the system stems from two problems: finding the key in a large key space and that of decoding a more or less arbitrary $[n, k]$-linear code with upto t errors.
If the first problem is successfully tackled i.e. recover the generator matrix $\mathbf{G}$ from $\mathbf{G}^{'}$, one can then use the fast decoding algorithm $\mathcal{D}$. However, this approach is extremely unlikely to succeed for large enough values of n and t, because there are just so many possibilities for $\mathbf{S}$, $\mathbf{G}$ and $\mathbf{P}$.
For a brute force approach to the second problem, general decoding problem for linear codes has been proved to be $NP-complete$ and so if code parameters are large enough, this attack will also be infeasible. A better approach would be to select k of the n coordinates randomly with the hope that none of them are error bits and based on this, find $\mathbf{u}$. The probability that none are error bits is

$$\left(1 - \frac{t}{n}\right)^k$$

and work involved in solving k simultaneous equations in k unknowns would be k^3 and hence, before $\mathbf{u}$ is found, one would expect a work factor of

$$k^3.\left(1 - \frac{t}{n}\right)^k$$

which for McEliece's suggested values $n = 1024$, $k = 524$, $t = 50$ would be $10^{19} \simeq 2^{65}$

Below are some results [2] with respect to some attacks on the McEliece Cryptosystem for the original parameters $n = 1024 = 2^{10}$ and $k = 524$ as suggested by McEliece :

- Guessing **S** and **P** : The probability of guessing the correct permutation matrix is $\frac{1}{1024!} \approx 0.18 \times 10^{-2639}$
- Exhaustive codeword comparison : Generate all 2^k codewords and find the unique codeword nearest to y. For the original parameters, this requires $2^{524} \approx 0.55 \times 10^{158}$
- Syndrome Decoding : For this as described above, the number of error vectors to try would be 0.33×10^{86}
- Bit Swapping : Number of swaps required $= 0.78 \times 10^{18}$

6 Conclusion

In this report, we presented the paradigm of cryptosystems using error correcting codes. We presented an overview of the McEliece cryptosystem which uses Goppa codes as the inherent linear error correcting codes. We gave a brief overview of the security of this system which is based on the hardness of decoding general linear codes.

References

[1] V. D. Goppa. A New Class of Linear Error Correcting Codes. *Probl. Pered. Inform.*, 6:24–30, September 1970.

[2] Ellen Jochemsz. Goppa Codes & the McEliece Cryptosystem. Master's thesis, Vrije Universiteit Amsterdam, 2002.

[3] R. J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Deep Space Network Progress Report*, 44:114–116, January 1978.

[4] N. J. Patterson. The Algebraic Decoding of Goppa Codes. *IEEE Trans. Information Theory*, 21:203–207, 1975.