Understanding binary Goppa decoding in Classic McEliece

Ganyu Xu[1]

University of Waterloo, Waterloo, Ontario, Canada g66xu@uwaterloo.ca

Abstract. Supplementary materials to "Understanding binary Goppa decoding" that reflect the actual decoder used in the reference implementations of Classic McEliece in NIST PQC Round 4 submission

Keywords: Error-correcting code · Classic McEliece · Post-quantum cryptography

1 Introduction

2 Preliminaries

3 Berlekamp-Massey Algorithm

The Berlekamp-Massey algorithm was first proposed by Elwin Berlekamp [1] for decoding BCH code and later extended by James Massey [4] to synthesize shortest LFSR. In this section, we will review the Berlekamp-Massey algorithm from the perspective of finding the shortest LFSR that generates a given finite sequence, including some key results that proved the optimality of the generated LFSR.

Let $\mathbb{F}$ be some finite field. A linear-feedback shift register (LFSR) of length L is parameterized by the connection coefficients $c_1, c_2, \ldots, c_L \in \mathbb{F}$ and the seed values $s_0, s_1, \ldots, s_{L-1} \in \mathbb{F}$. An LFSR outputs an infinite sequence $s_0, s_1, \ldots$ where the first L digits are the seed values, then for $j \geq L$:

$$s_j + \sum_{i=1}^{L} c_i s_{j-i} = 0$$

By convention, an empty LFSR with length 0 outputs an infinite sequence of all 0's. We further define the **connection polynomial** of an LFSR to be $C(x) = 1 + c_1 x + c_2 x^2 + \ldots + c_L x^L$. Note that there is no requirement for the connection coefficients to be non-zero, so the degree of the polynomial can be less than the actual length of the corresponding LFSR.

Let $\mathbf{s} = (s_0, s_1, \ldots) \in \mathbb{F}^\infty$ be some infinite sequence. Let $L_N(\mathbf{s})$ denote the length of the shortest LFSR that generates the first N digits of $\mathbf{s}$ (where there is no ambiguity we will simply use L_N for short). First observe that $L_N \leq N$ because one can always find an LFSR that generates the first N digits simply by

using these digits as the seed values. Second, observe that L_N is **monotonically non-decreasing** with respect to N: $L_{N+1} \geq L_N$. This makes sense because any LFSR that generates the first $N+1$ digits is also an LFSR that generates the first N digits, so the shortest LFSR that generates the first N digits cannot be longer than the shortest LFSR that generates the first $N+1$ digits.

A summary of the shortest LFSR synthesis algorithm, as presented in [4], can be found in Figure 1

Algorithm 1 Shortest LFSR synthesis

Input: $(s_0, s_1, \ldots, s_{N-1}) \in \mathbb{F}$
Output: The connection polynomial $C(x)$ and length l of a shortest LFSR

$C(x) \leftarrow 1, l \leftarrow 0, n \leftarrow 0, B(x) \leftarrow 1, d_m \leftarrow 1, y \leftarrow 1$
while $n < N$ **do**
  $d_n \leftarrow s_n + \sum_{i=1}^{l} c_i s_{n-i}$
  if $d_n = 0$ **then**
    $y \leftarrow y + 1$
  else if $2l > n$ **then**
    $C(x) \leftarrow C(x) - d_n d_m^{-1} x^y B(x)$
    $y \leftarrow y + 1$
  else
    $T(x) \leftarrow C(x)$
    $C(x) \leftarrow C(x) - d_n d_m^{-1} x^y B(x)$
    $l \leftarrow n + 1 - l$
    $B(x) \leftarrow T(x)$
    $d_m \leftarrow d_n$
    $y \leftarrow 1$
  end if
  $n \leftarrow n + 1$
end while
return $C(x), l$

Fig. 1. The Berlekamp-Massey algorithm iteratively constructs a shortest LFSR that generates some input sequence

The optimality of Algorithm 1 is proved in Theorem 1 below:

Theorem 1 (shortest LFSR synthesis). *Let* $\mathbf{s}$ *be an infinite sequence and* $N \geq 0$*. If there exists an LFSR with length* $L_N(\mathbf{s})$ *that generates* $s_0, s_1, \ldots, s_{N-1}$ *but not* s_N*, then* $L_{N+1}(\mathbf{s}) = \max(L_N(\mathbf{s}), N + 1 - L_N(\mathbf{s}))$

To prove Theorem 1, we will first prove a Lemma

Lemma 1 (LFSR length change). *If there exists an LFSR with length L that generates $s_0, s_1, \ldots, s_{N-1}$ but not s_N, then the length L' of any LFSR that generates $s_0, s_1, \ldots, s_N$ must be such that $L' \geq N + 1 - L$*

Proof. Let $(c_1, c_2, \ldots, c_L)$ be an length-L LFSR that generates $s_0, \ldots, s_{N-1}$ but not s_N. Let $(c'_1, c'_2, \ldots, c'_{L'})$ be some length L' LFSR that generates $s_0, \ldots, s_N$, then by the definition of LFSR we have the following three equations:

$$s_j = -\sum_{i=1}^{L} c_i s_{j-i} \qquad (L \leq j < N) \tag{1}$$

$$s_N \neq -\sum_{i=1}^{L} c_i s_{N-i} \tag{2}$$

$$s_j = -\sum_{i=1}^{L'} c'_i s_{j-i} \qquad (L' \leq j \leq N) \tag{3}$$

If $L' \leq N - L$, then $N - L \leq j \leq N$ falls in the range of equation 3. The RHS of equation 2 can be re-written:

$$\begin{aligned} -\sum_{i=1}^{L} c_i s_{N-i} &= -\sum_{i=1}^{L} c_i (-\sum_{k=1}^{L'} c'_k s_{N-i-k}) \\ &= -\sum_{k=1}^{L'} c'_k (-\sum_{i=1}^{L} c_i s_{N-i-k}) \\ &= -\sum_{k+1}^{L'} c_k^{L'} s_{N-k} \\ &= s_N \end{aligned}$$

This contradicts equation 2, thus it must be that $L' \geq N + 1 - L$.

Let $n < N$. Let $C^{(n)}(x)$ denote the connection polynomial of the shortest LFSR that generates the first n digits. If $C^{(n)}$ also generates s_n, then $C^{(n+1)} = C^{(n)}$ because of monotonicity. If $C^{(n)}$ does not generate s_n, then by Lemma 1 we have $L_{n+1} \geq n + 1 - L_n$. Furthermore because of monotonicity, $L_{n+1} \geq L_n$, thus we can establish a (recursive) lower bound : $L_{n+1} \geq \max(L_n, n + 1 - L_n)$.

It turns out that this lower bound can be achieved: suppose $C^{(n)}$ generates the first n digits but not the first $n + 1$ digits. Let $d_n = s_n + \sum_{i=1}^{L_n} c_i^{(n)} s_{n-i}$, by the earlier assumption we know $d_n \neq 0$. Let m be the number of digits such that $L_m < L_{m+1} = L_n$ (m is the sequence length before the last LFSR length change) and let $d_m = s_m + \sum_{i=1}^{L_m} c_i^{(m)} s_{m-i}$. Because $L_m < L_{m+1}$, d_m must also be non-zero. We claim that:

$$C^{(n+1)} = C^{(n)} - d_n d_m^{-1} x^{n-m} C^{(m)}$$

We need to verify two points: one is that $C^{(n+1)}$ indeed generates the first $n+1$ digits, second is that $C^{(n+1)}$ achieves the lower bound. First expand $C^{(n+1)}$:

$$\begin{aligned} C^{(n+1)} &= 1 + \sum_{i=1}^{L_n} c_i^{(n)} x^i - d_n d_m^{-1} x^{n-m} \left(1 + \sum_{i=1}^{L_m} c_i^{(m)} x^i\right) \\ &= 1 + \sum_{i=1}^{L_n} c_i^{(n)} x^i - d_n d_m^{-1} \left(x^{n-m} + \sum_{i=1}^{L_m} c_i^{(m)} x^{n-m+i}\right) \end{aligned}$$

Because of monotonicity we know $L_{n+1} \geq L_n, L_m$, so for $j \geq L_{n+1}$:

$$\begin{aligned} \text{j-th output} &= -\sum_{i=1}^{L_n} c_i^{(n)} s_{j-i} + d_n d_m^{-1} \left(s_{j-(n-m)} + \sum_{i=1}^{L_m} c_i^{(m)} s_{j-(n-m+i)}\right) \\ &= -\sum_{i=1}^{L_n} c_i^{(n)} s_{j-i} + d_n d_m^{-1} \left(s_{m-(j-n)} + \sum_{i=1}^{L_m} c_i^{(m)} s_{m-(j-n)+i)}\right) \end{aligned}$$

Where $j < n$, $-\sum_{i=1}^{L_n} c_i^{(n)} s_{j-i}$ evaluates to s_j, and $m - (j-n) < m$, so $s_{m-(j-n)} + \sum_{i=1}^{L_m} c_i^{(m)} s_{m-(j-n)+i} = 0$. The entire expression evaluates to s_j.

On the other hand, if $j = n$, then $-\sum_{i=1}^{L_n} c_i^{(n)} s_{j-i} = s_j - d_n$, and $m-(j-n) = m$, so $s_{m-(j-n)} + \sum_{i=1}^{L_m} c_i^{(m)} s_{m-(j-n)+i} = d_m$. Thus:

$$\text{j-th output} = (s_n - d_n) + d_n d_m^{-1} d_m = s_n$$

Hence $C^{(n+1)}$ generates the first $n+1$ digits. For the second point, observe that:

$$\begin{aligned} \deg(C^{(n+1)}) &= \deg(C^{(n)} - d_n d_m x^{n-m} C^{(m)}) \\ &= \max(\deg(C^{(n)}), \deg(x^{n-m} C^{(m)})) \\ &= \max(L_n, n - m + L_m) \end{aligned}$$

We inductively assume that the lower bound holds for all $j < n$, then $L_n = L_{m+1} = \max(L_m, m + 1 - L_m)$. Since we assumed $L_m < L_{m+1}$, it must be that $L_n = L_{m+1} = m + 1 - L_m$. Therefore:

$$\begin{aligned} \deg(C^{(n+1)}) &= \max(L_n, n - m + L_m) \\ &= \max(L_n, n - m + (m + 1 - L_n)) \\ &= \max(L_n, n + 1 - L_n) \end{aligned}$$

This proves that the lower bound can be achieved, which proves Theorem 1.

There are two more points to discuss regarding Algorithm 1. First, in the proof above we assumed that when $C^{(n)}$ fails to generate s_n, some $m : L_m < L_{m+1} < L_n$ exists, which is not the case for when s_n is the first non-zero digit of the sequence. In this case, the algorithm simply set the initial condition to be such that $C^{(n+1)}$ is correct, and after this iteration, m and its related parameters

$C^{(m)}(x)$ and d_m are guaranteed to be updated, so subsequent iterations will work according to the proof above.

Second, there is a distinction betweeen $2l > n$ and $2l \leq n$ because it decides whether there is going to be a length change, which will decide how $C^{(m)}, d_m, y$ will be updated. When $2l > n$, $l \geq n+1-l$, so $l \leftarrow \max(l, n+1-l) = l$ does not incur a length change; on the other hand, when $2l \leq n$, $l < n+1-l$, so there will be a length change, meaning that the current set of connection polynomial $C^{(n)}$ and difference d_n becomes "the most recent length change".

4 Alternant code decoding

5 Double-sized syndrome

Let $\mathbb{F}_{2^m}$ be some binary extension field, let $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{F}_{2^m}$ be n distinct elements, let $g(x) \in \mathbb{F}_{2^m}[x]$ be a monic irreducible polynomial of degree t. The binary Goppa code parameterized by $\Gamma = (\alpha_1, \ldots, \alpha_n, g)$ is defined by:

$$\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_2^n \mid \sum_{i=1}^{n} \frac{c_i A}{x - \alpha_i} \equiv 0 \mod g\}$$

Where $A = \prod_{i=1}^{n}(x - \alpha_i)$ is the vanishing polynomial of $\alpha_1, \ldots, \alpha_n$. Note that the original formulation of Goppa code only required $g \in \mathbb{F}_{2^m}[x]$ to be square-free and co-prime with A (i.e. $g(\alpha_i) \neq 0$ for all $1 \leq i \leq n$), but in Classic McEliece g is sampled to be monic irreducible, so we also require g to be monic ireducible.

Section 4 described how Algorithm 1 can be used to produce the error-locator polynomial $a(x)$ given a parity check matrix $H \in \mathbb{F}^{t \times n}$ and a received codeword $\mathbf{r} \in \mathbb{F}^n$, but this algorithm can only correct up to $\lceil \frac{t}{2} \rceil$ errors instead of the full-capacity t. In this section, we will review the techniques used in [3] to allow the application of Berlekamp-Massey algorithm for recovering all t errors.

Theorem 2 (Double-sized parity check matrix). *Given binary Goppa code $\Gamma = (\alpha_1, \ldots, \alpha_n, g)$ over binary extension field $\mathbb{F}_{2^m}$, $\mathbf{c} \in \mathbf{F}_2^n$ is a codeword if and only if for all $0 \leq s \leq 2t-1$:*

$$\sum_{i=1}^{n} \frac{\alpha_i^s c_i}{g^2(\alpha_i)} = 0$$

Theorem 2 provides an alternative parity check matrix $H^{(2)}$ that has $2t$ rows instead of t rows as in the canonical parity check matrix. For the remainder of this section we will prove Theorem 2 using a series of Theorems and Lemmas.

Proof. We begin by stating a well-known results:

Theorem 3 (Goppa squaring). *Given binary extension field $\mathbb{F}_{2^m}$, n distinct elements $\alpha_1, \ldots, \alpha_n$, and degree-$t$ monic irreducible polynomial $g(x) \in \mathbb{F}_{2^m}[x]$, $\sum_{i=1}^{n} \frac{c_i A}{x-\alpha_i} \equiv 0 \mod g$ if and only if $\sum_{i=1}^{n} \frac{c_i A}{x-\alpha_i} \equiv 0 \mod g^2$*

See [2] for proof.

Let $B = \sum_{i=1}^{n} \frac{c_i A}{g^2(\alpha_i)(x-\alpha_i)}$ and $C = \sum_{i=1}^{n} \frac{c_i A}{x-\alpha_i}$, we claim:

$$C \equiv 0 \mod g^2 \iff \deg(B) < n - 2t \tag{4}$$

Observe that for each $1 \leq j \leq n$, $B(\alpha_j)g^2(\alpha_j) = C(\alpha_j)$, which means that $\alpha_1, \alpha_2, \ldots, \alpha_n$ are roots of the polynomial $Bg^2 - C$. Consequently $Bg^2 - C$ is divisible by A.

If $C \equiv 0 \mod g^2$, then $Bg^2 - C = g^2(B - C/g^2)$ where $B - C/g^2 \in \mathbb{F}_{2^m}[x]$. Because g and A are co-prime, it must be that A divides $B - C/g^2$, but the degree of B and the degree of C/g^2 are both less than n, so it must be that $B - C/g^2 = 0$. It is easy to show that $\deg(C) \leq n-1$, so $\deg(B) = \deg(C/g^2) \leq n - 1 - 2t < n - 2t$.

If $\deg(B) < n - 2t$, then $\deg(Bg^2) < n$, so $\deg(Bg^2 - C) < n$. But since A divides $Bg^2 - C$, it must be that $Bg^2 - C = 0$, which implies that $C = Bg^2$ is divisible by g^2.

Now denote $Q = \sum_{i=1}^{n} \frac{\alpha_i^{2t} c_i A}{g^2(\alpha_i)(x-\alpha_i)}$. It is easy to show the follwing:

$$\deg(B) < n - 2t \iff \deg(x^{2t}B - Q) < n \tag{5}$$

On the RHS of the relation above:

$$\begin{aligned} x^{2t}B - Q &= \sum_{i=1}^{n} \frac{(x^{2t} - \alpha_i^{2t}) c_i A}{g^2(\alpha_i)(x - \alpha_i)} \\ &= \sum_{i=1}^{n} \left(\frac{c_i A}{g^2(\alpha_i)} \sum_{s=0}^{2t-1} x^s \alpha_i^{2t-1-s} \right) \\ &= A \sum_{s=0}^{2t-1} \left(\sum_{i=1}^{n} \left(\frac{c_i \alpha_i^{2t-1-s}}{g^2(\alpha_i)} \right) x^s \right) \end{aligned}$$

In other words, $x^{2t}B - Q$ is divisible by A, so $\deg(x^{2t}B - Q) < n$ if and only if $x^{2t}B - Q = 0$, which happens if and only if $\sum_{i=1}^{n} \frac{c_i \alpha_i^{2t-1-s}}{g^2(\alpha_i)} = 0$ for all $0 \leq s \leq 2t-1$:

$$\deg(x^{2t}B - Q) < n \iff \sum_{i=1}^{n} \frac{c_i \alpha_i^s}{g^2(\alpha_i)} = 0 \ \ (0 \leq s \leq 2t-1) \tag{6}$$

Combining Theorem 3 and equations 4, 5, 6 proves Theorem 2.

References

1. Berlekamp, E.R.: Nonbinary BCH decoding (abstr.). IEEE Trans. Inf. Theory **14**(2), 242 (1968). https://doi.org/10.1109/TIT.1968.1054109, https://doi.org/10.1109/TIT.1968.1054109
2. Bernstein, D.J.: Understanding binary-goppa decoding. IACR Cryptol. ePrint Arch. p. 473 (2022), https://eprint.iacr.org/2022/473

3. Heyse, S., Güneysu, T.: Code-based cryptography on reconfigurable hardware: tweaking niederreiter encryption for performance. J. Cryptogr. Eng. **3**(1), 29–43 (2013). https://doi.org/10.1007/S13389-013-0056-4, https://doi.org/10.1007/s13389-013-0056-4
4. Massey, J.L.: Shift-register synthesis and BCH decoding. IEEE Trans. Inf. Theory **15**(1), 122–127 (1969). https://doi.org/10.1109/TIT.1969.1054260, https://doi.org/10.1109/TIT.1969.1054260