Berlekamp-Massey Algorithm: Euclid in Disguise

Ishai Ilani
Western Digital
Kfar Saba, Israel
Email: ishai.ilani@wdc.com

Abstract*—In this paper we take a fresh look at the well-known Berlekamp-Massey (BM) algorithm for decoding of Reed-Solomon (RS) and Bose–Chaudhuri–Hocquenghem (BCH) codes. RS and BCH codes are a very important family of cyclic codes, and are included in most elementary courses on code theory. One of the most important tools in decoding of RS and BCH codes was developed by Berlekamp, and later formulated as an algorithm for synthesizing short LFSR-s by Massey and is now known as the Berlekamp-Massey (BM) algorithm. An alternative algorithm for decoding such codes is the extended Euclid algorithm. We present another viewpoint to the BM algorithm which is simpler than the Massey formulation, and mirrors the extended Euclid algorithm. This presentation may replace the common treatment of the BM algorithm in elementary courses with a simpler and more rigorous presentation. Moreover, this approach enables to improve the HW implementation of BCH decoding. This is promising as BCH codes are gaining renewed interest lately in latency sensitive applications. Another advantage of the new approach is that it provides a simple derivation of erasure decoding.

Keywords—BM algorithm; Euclid algorithm; backward polynomial division; backward Euclid algorithm; ged; dom;

I. INTRODUCTION

The classic way of decoding BCH codes is a three step algorithm. During the first step the syndromes are computed. The second step computes an Error Locator Polynomial (ELP), and the third step finds the roots of the ELP from which the error locations may be deduced. The second step may be solved by an extended Euclid algorithm introduced, by [10] or by the Berlekamp Massey (BM) algorithm (first presented in [1] and [2]).

Ever since their publication, various authors have attempted to find simpler derivations of Massey's proofs (e.g. [5]), and exploit the similarities between the BM and Euclid approaches, (cf. [6], [7], [8], and [9]). In some sense this paper follows the path of these papers, and finds another simplified proof of the BM algorithm, and similarity between the BM and extended Euclid algorithms. In this paper we go a step further and introduce the concepts of *backward polynomial division* and *backward Euclid algorithm*. This enables a full derivation of the ELP in a conceptual and natural language. In a leisure pace, we are able to introduce the new concepts, provide a natural derivation of the key equation, and reconstruct the claims and proofs of [2] in 5 pages and in a language suitable for undergraduate students. Moreover our approach provides a simple proof that the BM algorithm applied to BCH codes requires only half the number of iterations than a BM algorithm applied to RS codes. Also our treatment of the uniqueness of the *minimal degree solution* is complete and accurate. Therefore we believe that the presentation in this paper may replace the common presentation of the subject in textbooks, (e.g. [4]), and courses on algebraic codes. For simplicity we limit the presentation to codes whose parity check matrix $\boldsymbol{H}$ may be written as:

$$\boldsymbol{H} = \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdots & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^{2\cdot 2} & \cdots & \cdots & \alpha^{(n-1)\cdot 2} \\ 1 & \alpha^3 & \alpha^{3\cdot 2} & \cdots & \cdots & \alpha^{(n-1)\cdot 3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{2t} & \alpha^{2t\cdot 2} & \cdots & \cdots & \alpha^{(n-1)\cdot 2t} \end{pmatrix}, \quad (1)$$

where α is a primitive element of $GF(2^m)$ for some integer m, $n = 2^m$-1 is the length of the code, and the distance of the code is greater than $2t$. The columns of $\boldsymbol{H}$ are indexed from 0 to n-1.

II. A NOTE ON NOTATION

Our conventions for notation are as follows: Bold upper case will be used for the single matrix $\boldsymbol{H}$ which appears in this paper.

Galois field scalars will be denoted by lowercase letters, and α will denote a primitive element of the Galois field. Due to common conventions integer scalars will be denoted by lower and upper case letters. This should not cause any confusion as it shall be clear from the context when a lower case letter refers to a Galois field element or an integer.

Vectors and polynomials will be denoted by lower case bold letters, and unknown polynomials will be denoted by $\boldsymbol{x}, \boldsymbol{y}$. An exception are the basis polynomials which have the special notation $\mathfrak{G}$ and $\mathfrak{H}$.

Usually the index of coefficients of polynomials will match the degree of their corresponding monomial. The syndrome polynomial $\boldsymbol{s}$ will be a slight variant and will expand as

$$\boldsymbol{s}(D) = \sum_{i=1}^{deg(\boldsymbol{s})+1} s_i D^{i-1}. \quad (2)$$

The reason for this exception is that using this notation the equation $s_{2i} = s_i^2$ holds for BCH codes.

III. NATURAL DERIVATION OF THE KEY EQUATION

Suppose a received word has an error vector $\boldsymbol{e} = (e_0, e_1, \ldots, e_{n-1})^T$ with support at $i(1), i(2), \ldots, i(L) \in [0, n-1]$, i.e. the received word has L errors at coordinates $i(1), i(2), \ldots, i(L)$, whose values are $e_{i(1)}, e_{i(2)}, \ldots, e_{i(L)}$. The syndrome vector $\boldsymbol{s} = (s_1, s_2, \ldots, s_{2t})^T$ defined as

$$\boldsymbol{s} = \boldsymbol{H}\boldsymbol{e}\ , \quad (3)$$

978-1-5386-6378-3/18/$31.00 ©2018 IEEE

is a linear combination of the $i(1), i(2), \ldots, i(L)$ columns of the matrix $\boldsymbol{H}$, where each of the columns is a *geometric progression*. The decoding problem may be directly translated to the following statement

$$\begin{gathered}\textit{find } L \le t \textit{ geometric progressions such that} \\ \boldsymbol{s} \textit{ is a linear combination of them.}\end{gathered} \tag{4}$$

Actually (4) suggests that the decoding problem may be generalized to a pure mathematical problem over any field: Given a vector $\boldsymbol{s}$ in a $2t$ dimensional vector space, is $\boldsymbol{s}$ a sum of $\le t$ geometric progressions? If so find the quotients and first elements of the progressions.

We shall solve the problem indirectly by combining properties of geometric progressions with a uniqueness theorem. Transforming $\boldsymbol{s}$ and the columns of $\boldsymbol{H}$ to a polynomial representation and using the well-known formula of a sum of a geometric progression we get

$$\begin{gathered}\boldsymbol{s}(D) \triangleq^{\text{def}} \sum_{i=1}^{2t} s_i D^{i-1} = \\ \sum_{l=1}^{L} e_{i(l)} \alpha^{i(l)} \left[\sum_{j=0}^{2t-1} \left(\alpha^{i(l)} D\right)^j\right] = . \\ \sum_{l=1}^{L} e_{i(l)} \alpha^{i(l)} \frac{1-\left(\alpha^{i(l)} D\right)^{2t}}{1-\alpha^{i(l)} D}\end{gathered} \tag{5}$$

The *Error Locator Polynomial*, (ELP), is defined naturally as the common denominator of (5) and given explicitly by

$$\boldsymbol{c}(D) = \prod_{l=1}^{L} \left(1 - \alpha^{i(l)} D\right). \tag{6}$$

Expanding (6) $\boldsymbol{c}(D)$ may be written as $\boldsymbol{c}(D) = \sum_{i=0}^{L} c_i D^i$, with $c_0 = 1$. Each of numerators in the last line of (5) taken modulo D^{2t} is a scalar. Multiplying by $\boldsymbol{c}(D)$ we get

$$\boldsymbol{s}(D)\boldsymbol{c}(D) \bmod D^{2t} = \boldsymbol{v}(D), \tag{7}$$

where $\boldsymbol{v}(D)$ given by

$$\boldsymbol{v}(D) = \sum_{l=1}^{L} \frac{e_{i(l)} \alpha^{i(l)}}{1-\alpha^{i(l)} D} \boldsymbol{c}(D), \tag{8}$$

is called the *error evaluator polynomial*. Since each of the denominators in (8) divides $\boldsymbol{c}(D)$ we see that $deg(\boldsymbol{v}) < deg(\boldsymbol{c})$. Combining (5),(6), (7), (8) implies that setting $\boldsymbol{x} = \boldsymbol{c}(D)$, $\boldsymbol{y} = \boldsymbol{v}(D)$ the pair $(\boldsymbol{x}, \boldsymbol{y})$ is a solution of the equation

$$\boldsymbol{x}\boldsymbol{s}(D) = \boldsymbol{y} \bmod D^{2t} \tag{9}$$

with c_0=1 and $deg(\boldsymbol{v}) < deg(\boldsymbol{c}) = L \le t$. Equation (9) is known as the *key equation* in the unknown polynomials $\boldsymbol{x}, \boldsymbol{y}$.

Actually it is more accurate to write (9) as $\boldsymbol{y} = \boldsymbol{x}\boldsymbol{s}(D) \bmod D^{2t}$, but (9) is the common way the *key equation* appears in the literature, so we also adopt this form.

IV. Solving the Key Equation - Theory

Consider a slight modification of equation (9)

$$\boldsymbol{x}D\boldsymbol{s}(D) - \boldsymbol{y} = 0 \bmod D^N. \tag{10}$$

Definitions: 1. For a pair of polynomials $(\boldsymbol{c}, \boldsymbol{w})$ define the degree of the pair as $deg(\boldsymbol{c}, \boldsymbol{w}) \triangleq^{\text{def}} max\{deg(\boldsymbol{c}), deg(\boldsymbol{w})\}$. This definition extends naturally to any vector of polynomials. 2. A solution $(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{c}, \boldsymbol{w})$ of (10) with $c_0 \neq 0$, is called a *minimal degree solution* if any solution $(\boldsymbol{c}', \boldsymbol{w}')$ of (10) with $c'_0 \neq 0$ satisfies $deg(\boldsymbol{c}', \boldsymbol{w}') \ge deg(\boldsymbol{c}, \boldsymbol{w})$.

For $N = 2t + 1$ The solutions of (9) and (10) are closely related. If $(\boldsymbol{c}, \boldsymbol{v})$ solves (9) with $deg(\boldsymbol{v}) < t$, $deg(\boldsymbol{c}) \le t$ then $(\boldsymbol{c}, \boldsymbol{w}) \triangleq^{\text{def}} (\boldsymbol{c}, D\boldsymbol{v})$ solves (10) with $deg(\boldsymbol{c}, \boldsymbol{w}) \le t$, and conversely if $(\boldsymbol{c}, \boldsymbol{w})$ with $deg(\boldsymbol{c}, \boldsymbol{w}) \le t$ solves (10) then $(\boldsymbol{c}, \boldsymbol{v}) \triangleq^{\text{def}} (\boldsymbol{c}, \boldsymbol{w}/D)$ solves (9) with $deg(\boldsymbol{v}) < t$, $deg(\boldsymbol{c}) \le t$.

For a syndrome polynomial $\boldsymbol{s}$ we shall solve (10) for $N = 2t + 1$ by an iterative algorithm comprising of $2t$ iterations, where at iteration j the algorithm computes a pair of polynomials $\boldsymbol{c}^{(j)}$, and $\boldsymbol{w}^{(j)}$ with $c_0^{(j)} \neq 0$ such that $(\boldsymbol{c}^{(j)}, \boldsymbol{w}^{(j)})$ solves (10) for $N = j + 1$. Moreover this solution is a *minimal degree solution*. Therefore the pair $(\boldsymbol{c}, \boldsymbol{w}) \triangleq^{\text{def}} \left(\boldsymbol{c}^{(2t)}, \boldsymbol{w}^{(2t)}\right)$ is a *minimal degree solution* of (10) for $N = 2t + 1$ with $c_0 \neq 0$.

If $\boldsymbol{s}$ is the polynomial associated with (3) and $L \le t$, then the pair $(\boldsymbol{c}(D), D\boldsymbol{v}(D))$ where $\boldsymbol{c}(D)$ is given by (6) and $\boldsymbol{v}(D)$ is given by (8) is also a *minimal degree solution* with the properties c_0=1, and $deg(\boldsymbol{v}(D)) < deg(\boldsymbol{c}(D)) \le t$. A uniqueness theorem will come to the rescue and state that if $L \le t$ the *key equation* has at most one *minimal degree solution* (up to scalar multiplication), therefore $\boldsymbol{c}^{(2t)}$ is the desired ELP, and $\boldsymbol{v}^{(2t)} \triangleq^{\text{def}} \boldsymbol{w}^{(2t)}/\mathrm{D}$ is the *error evaluator polynomial*.

V. Minimal Degree Solutions Of The Key EQUation

Lemma 1: Let $(\boldsymbol{c}, \boldsymbol{w})$ be a minimal degree solution of equation (10) with $c_0 \neq 0$. Then $\boldsymbol{c}$ and $\boldsymbol{w}$ are relatively prime.

Proof: Suppose not. Then $\boldsymbol{c} = \boldsymbol{p}\boldsymbol{c}'$, $\boldsymbol{w} = \boldsymbol{p}\boldsymbol{w}'$ for some polynomial $\boldsymbol{p}$ satisfying $deg(\boldsymbol{p}) > 0$. Satisfying (10) implies that $\boldsymbol{c}D\boldsymbol{s}(D) - \boldsymbol{w} = D^N\boldsymbol{q}$ for some polynomial $\boldsymbol{q}$. So $\boldsymbol{p}\boldsymbol{c}'D\boldsymbol{s}(D) - \boldsymbol{p}\boldsymbol{w}' = D^N\boldsymbol{q}$. The polynomial $\boldsymbol{p}$ must divide $\boldsymbol{q}$ (the free variable of $\boldsymbol{p}$ is non-0 so $\boldsymbol{p}$ is prime to D^N), so $(\boldsymbol{c}', \boldsymbol{w}')$ satisfies $\boldsymbol{c}'D\boldsymbol{s}(D) - \boldsymbol{w}' = 0 \bmod D^N$ and $c'_0 \neq 0$ contradicting the minimality of $(\boldsymbol{c}, \boldsymbol{w})$.

Lemma 2: Any solution $(\boldsymbol{c}', \boldsymbol{w}')$ of (10) with $deg(\boldsymbol{c}', \boldsymbol{w}') < N/2$, and $c'_0 \neq 0$, satisfies $(\boldsymbol{c}', \boldsymbol{w}') = (\boldsymbol{p}\boldsymbol{c}, \boldsymbol{p}\boldsymbol{w})$ for a minimal degree solution $(\boldsymbol{c}, \boldsymbol{w})$, where $\boldsymbol{p}$ is a polynomial.

Proof: Let $(\boldsymbol{c}, \boldsymbol{w})$ be a minimal solution of (10). Then

$$\boldsymbol{c}(\boldsymbol{c}'D\boldsymbol{s} - \boldsymbol{w}') = 0 \bmod D^N \cdot \tag{11}$$

Similarly

$$\boldsymbol{c}'(\boldsymbol{c}D\boldsymbol{s} - \boldsymbol{w}) = 0 \bmod D^N. \tag{12}$$

Subtract to get

$$\boldsymbol{c}'\boldsymbol{w} - \boldsymbol{c}\boldsymbol{w}' = 0 \bmod D^N. \tag{13}$$

According to the assumptions $deg(\boldsymbol{c}'\boldsymbol{w} - \boldsymbol{c}\boldsymbol{w}') < N$ thus

$$\boldsymbol{c}'\boldsymbol{w} = \boldsymbol{c}\boldsymbol{w}' \Rightarrow \boldsymbol{c}' = \boldsymbol{c}\boldsymbol{w}'/\boldsymbol{w}. \tag{14}$$

According to Lemma 1 $\boldsymbol{c}$ and $\boldsymbol{w}$ are relatively prime, so $\boldsymbol{w}$ must divide $\boldsymbol{w}'$ to generate a polynomial $\boldsymbol{p} = \boldsymbol{w}'/\boldsymbol{w}$, from which we may immediately deduce $(\boldsymbol{c}', \boldsymbol{w}') = (\boldsymbol{p}\boldsymbol{c}, \boldsymbol{p}\boldsymbol{w})$.

Corollary: If $\boldsymbol{s}$ is a syndrome vector associated with the code defined by (1), and $\boldsymbol{s}$ is a sum of $L \le t$ geometric progressions then the polynomial representation of $\boldsymbol{s}$ together with the polynomial pair comprised of $\boldsymbol{c}(D)$ of (6) and $D\boldsymbol{v}(D)$, where $\boldsymbol{v}(D)$ is given by (8) is a polynomial multiple of the minimal solution of (10) with $N = 2t + 1$. However, if some proper divisor of this solution is also a solution, then the progressions whose quotients are associated with the roots of the divisor will not be part of the sum. Subtracting one solution from the other results in a linear dependence between $L \le t$ geometric progressions of length $2t$ which is impossible. Therefore, an

algorithm that converges to a minimal degree solution will converge to the polynomial pair comprising the ELP and the *evaluator polynomial*, (or a scalar multiple of this pair). In this case the number of roots of $\boldsymbol{c}(D)$ will be equal to its degree.

On the other hand, suppose $\boldsymbol{s}$ may not be written as a sum of $\leq t$ geometric progressions. In this case there are 2 possibilities. Either the algorithm will converge to a solution whose degree is larger than t or the algorithm will converge to a minimal solution $(\boldsymbol{c},\boldsymbol{w})$ with the desired conditions, (i.e. $c_0 \neq 0,\ \ deg(\boldsymbol{c},\boldsymbol{w}) \leq t$). In the latter case the number of roots of $\boldsymbol{c}$ will be strictly less than its degree as such a $\boldsymbol{c}$ would not be a product of $deg(\boldsymbol{c})$ different linear factors. This completes a full and rigorous treatment of this issue, which is not presented accurately in neither [2] nor [4].

VI. Backward Polynomial Division

For a polynomial $\boldsymbol{g}(D)$ in the variable D, let $ged(\boldsymbol{g})$ denote the lowest index of the non-zero coefficients of $\boldsymbol{g}$, (mirroring $deg(\boldsymbol{g})$ which is the highest such index). Using this terminology any polynomial $\boldsymbol{g}(D)$ may be written as

$$\boldsymbol{g}(D) = \sum_{i=ged(\boldsymbol{g})}^{deg(\boldsymbol{g})} g_i D^i \tag{15}$$

where both $g_{ged(\boldsymbol{g})}$ and $g_{\deg(\boldsymbol{g})}$ are non-zero elements.

For 2 polynomials $\boldsymbol{g}(D)$ and $\boldsymbol{h}(D)$ with $deg(\boldsymbol{g}) \geq deg(\boldsymbol{h})$ we are familiar with the long division and the pair of quotient and remainder polynomials $(\boldsymbol{q}(D), \boldsymbol{r}(D))$ satisfying

$$\begin{matrix} \boldsymbol{g} = \boldsymbol{q}\boldsymbol{h} + \boldsymbol{r} \\ deg(\boldsymbol{r}) < deg(\boldsymbol{h}) \end{matrix}. \tag{16}$$

The process of computing $(\boldsymbol{q},\boldsymbol{r})$ is well-known and one variant suited for HW implementation is presented in (17).

$$\begin{aligned} &\boldsymbol{r} = \boldsymbol{g},\ \ \boldsymbol{q} = 0, h = h_{deg(\boldsymbol{h})}, i = deg(\boldsymbol{g}) \\ &Loop\ until\ deg(\boldsymbol{r}) < deg(\boldsymbol{h}) \\ &\quad \tilde{\boldsymbol{q}} = \tfrac{r_{\mathrm{i}}}{h} D^{[i-deg(\boldsymbol{h})]} \\ &\quad \boldsymbol{r} = \boldsymbol{r} - \tilde{\boldsymbol{q}}\boldsymbol{h} \\ &\quad \boldsymbol{q} = \boldsymbol{q} + \tilde{\boldsymbol{q}} \\ &\quad i = i - 1 \end{aligned} \tag{17}$$

The remainder $\boldsymbol{r}$ is known as $\boldsymbol{g}\ mod\ \boldsymbol{h}$.

Let $\boldsymbol{g}$ and $\boldsymbol{h}$ be 2 polynomials such that $ged(\boldsymbol{g}) \geq ged(\boldsymbol{h})$. Define a *backward polynomial division* (BPD) process by the following sequence of polynomials

$$\begin{aligned} &\boldsymbol{r} = \boldsymbol{g},\ \ \boldsymbol{q} = 0, h = h_{ged(\boldsymbol{h})}, i = ged(\boldsymbol{g}) \\ &Loop\ until\ stopping\ condition\ is\ satisfied \\ &\quad \tilde{\boldsymbol{q}} = \tfrac{r_{\mathrm{i}}}{h} D^{[i-ged(\boldsymbol{h})]} \\ &\quad \boldsymbol{r}_{prev} = \boldsymbol{r} \\ &\quad \boldsymbol{r} = \boldsymbol{r} - \tilde{\boldsymbol{q}}\boldsymbol{h} \\ &\quad \boldsymbol{q} = \boldsymbol{q} + \tilde{\boldsymbol{q}} \\ &\quad i = i + 1 \end{aligned} \tag{18}$$

This process is the mirror of the ordinary process used for long polynomial division. Other than the obvious changes, (i.e. replacing *deg* by *ged*, and replacing i=$i-1$ by i=i+1), another change is the inclusion of the line $\boldsymbol{r}_{prev} = \boldsymbol{r}$. This change is not required for performing one instance of BPD. However, it is required when BPD is performed as part of a *backward Euclid algorithm*.

The ordinary polynomial division has a natural stopping condition when $deg(\boldsymbol{r}) < deg(\boldsymbol{h})$. For BPD the stopping point will be defined in the next section.

We denote the *"remainder"* $\boldsymbol{r}$ by a special notation

$$\boldsymbol{g}\ dom\ \boldsymbol{h} \stackrel{\text{def}}{=} \boldsymbol{r}, \tag{19}$$

which mirrors $\boldsymbol{g}\ mod\ \boldsymbol{h}$ of the ordinary polynomial division.

VII. Stopping Condition and Example

Consider the (basis) polynomials $\mathfrak{G},\mathfrak{H}$. For any polynomial $\boldsymbol{r}$ which is a polynomial combination of the basis polynomials, i.e. $\boldsymbol{r} = \boldsymbol{p}\mathfrak{G} + \boldsymbol{q}\mathfrak{H}$, we may associate $\boldsymbol{r} \leftrightarrow (\boldsymbol{p},\boldsymbol{q})$. The vector $(\boldsymbol{p},\boldsymbol{q})$ is called a Bezout pair, and denoted $(\boldsymbol{p},\boldsymbol{q}) = bp(\boldsymbol{r})$. For example $(1,0) = bp(\mathfrak{G}),\ \ (0,1) = bp(\mathfrak{H})$.

Bezout pairs are not unique, but if $\mathfrak{G},\mathfrak{H}$ are relatively prime, then for any $\boldsymbol{r}$ there is at most 1 Bezout pair $(\boldsymbol{p},\boldsymbol{q}) = bp(\boldsymbol{r})$, such that $deg(\boldsymbol{p},\boldsymbol{q}) < deg(\mathfrak{G},\mathfrak{H})$. We shall call such a Bezout pair, (if it exists), a minimal Bezout pair.

Back to the BPD algorithm and stopping condition. Given $\boldsymbol{g},\boldsymbol{h}$ and a pair of basis polynomials $\mathfrak{G},\mathfrak{H}$ perform the algorithm (18) and compute the minimal Bezout pairs of $\boldsymbol{g},\boldsymbol{h},\boldsymbol{r}$ relative to the basis polynomials $\mathfrak{G},\mathfrak{H}$. The stopping condition will be when

$$deg\big(bp(\boldsymbol{r})\big) > deg(bp(\boldsymbol{g})). \tag{20}$$

For example, consider the polynomials $\boldsymbol{g} = D^4 + D^2 + D + 1$, and $\boldsymbol{h} = D + 1$, with $\mathfrak{G} = \boldsymbol{g}, \mathfrak{H} = \boldsymbol{h}$. Applying a first iteration of (18) we get $\boldsymbol{r} = D^4 + D^2$, and $bp(\boldsymbol{r}) = (1,-1)$, thus $deg(bp(\boldsymbol{r})) = deg(bp(\boldsymbol{g})) = 0$. Applying a second iteration we get $\boldsymbol{r} = D^4 - D^3$, and $bp(\boldsymbol{r}) = (1,-(D^2+1))$. At this point $deg\big(bp(\boldsymbol{r})\big) > deg(bp(\boldsymbol{g}))$, and the BPD process stops.

In the next section we extend the BPD to a *backward Euclid algorithm* (BEA) as a mirror algorithm of the ordinary Euclid algorithm, and in section IX we apply the BEA to the polynomials $\boldsymbol{g} = D\boldsymbol{s}$, $\boldsymbol{h} = 1 + D\boldsymbol{s}$, with Bezout pairs computed relative to $\mathfrak{G} = \boldsymbol{g}$, $\mathfrak{H} = -1$, and prove that the Bezout pairs of $\boldsymbol{h}^{(i)}$ converge to a minimal degree solution of (10) with $N = 2t + 1$.

VIII. Backward Euclid Algorithm

Given basis polynomials $\mathfrak{G},\mathfrak{H}$, and 2 polynomials $\boldsymbol{g}$ and $\boldsymbol{h}$ expressed as polynomial combinations of $\mathfrak{G},\mathfrak{H}$ with $ged(\boldsymbol{g}) \geq ged(\boldsymbol{h}))$, $deg\big(bp(\boldsymbol{g})\big) \geq deg\big(bp(\boldsymbol{h})\big)$ define a *backward Euclid algorithm* (BEA) according to the following:

$$\begin{aligned} &\boldsymbol{g}^{(0)} = \boldsymbol{g},\ \ \boldsymbol{h}^{(0)} = \boldsymbol{h} \\ &\boldsymbol{g}^{(i)} = \boldsymbol{r}^{(i-1)} = \boldsymbol{g}^{(i-1)}\ dom\ \boldsymbol{h}^{(i-1)} \\ &\boldsymbol{h}^{(i)} = \boldsymbol{r}_{prev}^{(i-1)} \end{aligned} \tag{21}$$

Note that a solution $(\boldsymbol{c},\boldsymbol{w})$ of equation (10) may be interpreted as a Bezout pair of $\boldsymbol{c}D\boldsymbol{s}(D) - \boldsymbol{w}$ relative to the basis $\mathfrak{G} = D\boldsymbol{s}, \mathfrak{H} = -1$, and the challenge of solving the key equation is finding a solution of (10) with large ged, (=large N), and small degree of the associated Bezout pair. In this context the definition of BPD and BEA is quite natural, as in each step of the BEA $\boldsymbol{h}^{(i)}$ and $\boldsymbol{g}^{(i)}$ are the two polynomials with highest ged, and relatively low degree of Bezout pairs.

The BEA is a mirror process with a slight variation on the ordinary Euclid algorithm which operates according to

$$\begin{aligned} &\boldsymbol{g}^{(0)} = \boldsymbol{g}, \qquad \boldsymbol{h}^{(0)} = \boldsymbol{h} \\ &\boldsymbol{g}^{(i)} = \boldsymbol{h}^{(i-1)} \\ &\boldsymbol{h}^{(i)} = \boldsymbol{r}^{(i-1)} = \boldsymbol{g}^{(i-1)} \; mod \; \boldsymbol{h}^{(i-1)} \end{aligned} \tag{22}$$

The following table summarizes the BPD and BEA concepts and terminology in comparison with the ordinary polynomial division and Euclid algorithm.

TABLE I

Ordinary Division and Euclid	**Backward Division and Euclid**
Stopping Condition $deg(\boldsymbol{r}) < deg(\boldsymbol{h})$	Stopping Condition $deg(bp(\boldsymbol{r})) > deg(bp(\boldsymbol{g}))$
Euclid Remainder $\boldsymbol{r} = \boldsymbol{g} \; mod \; \boldsymbol{h}$	Euclid Remainder $\boldsymbol{r} = \boldsymbol{g} \; dom \; \boldsymbol{h}$
Euclid Update $\boldsymbol{g} \leftarrow \boldsymbol{h} \leftarrow \boldsymbol{r}$	Euclid Update $\boldsymbol{r} \rightarrow \boldsymbol{g}, \; \boldsymbol{r}_{prev} \rightarrow \boldsymbol{h}$
$deg(\boldsymbol{g}) \geq deg(\boldsymbol{h}) > deg(\boldsymbol{r})$	$ged(\boldsymbol{r}) > ged(\boldsymbol{g}) \geq ged(\boldsymbol{h})$

IX. BM AS BACKWARD EUCLID

The (ordinary) Euclid algorithm for computing the ELP of an RS code applies the Euclid algorithm to the pair

$$\boldsymbol{g} = D^{2t}, \; \boldsymbol{h} = \boldsymbol{s}, \tag{23}$$

(with $\mathfrak{G} = \boldsymbol{g}$, $\mathfrak{H} = \boldsymbol{h}$) and it terminates at the first iteration i for which $deg(\boldsymbol{g}^{(i)} \; mod \; \boldsymbol{h}^{(i)}) < t$. At this point denote

$$\begin{aligned} \boldsymbol{v} &= \boldsymbol{g}^{(i)} \; mod \; \boldsymbol{h}^{(i)} \\ (\boldsymbol{a}, \boldsymbol{c}) &= bp(\boldsymbol{v}) \end{aligned}, \tag{24}$$

$\boldsymbol{c}$ is the ELP and $\boldsymbol{v}$ is the evaluator polynomial.

A mirror computation may be done by the BEA.

Main proposition: If $\boldsymbol{s}$ is a syndrome polynomial of degree $2t$-1 associated with an error vector of weight $L \leq t$ of the code defined by (1), and the BEA is applied to $\boldsymbol{g}=D\boldsymbol{s}$, $\boldsymbol{h}=1+D\boldsymbol{s}$ with Bezout pairs computed relative to the basis polynomials $\mathfrak{G}=\boldsymbol{g}$,$\mathfrak{H}=-1$, then the Bezout pair $(\boldsymbol{c}^{(i)}, \boldsymbol{w}^{(i)})$ of the first $\boldsymbol{h}^{(i)}$ satisfying $ged(\boldsymbol{h}^{(i)}) > 2t$ is equal to the pair $(c_0^{(i)}\boldsymbol{c}(D), c_0^{(i)}D\boldsymbol{v}(D))$, where $\boldsymbol{c}(D)$ is the ELP of (6), and $\boldsymbol{v}(D)$ is the evaluator polynomial of (8).

Proof: According to the assumptions the polynomial pair comprised of $\boldsymbol{c}(D)$ of (6) and $D\boldsymbol{v}(D)$, where $\boldsymbol{v}(D)$ is given by (8) is a solution with degree $\leq t$ of (10) for $N=2t+1$. According to the corollary to Lemma 2 it is the unique (up to scalar multiplication) minimal degree solution of (10) for $N=2t+1$. Therefore it suffices to prove that for every $N \leq 2t+1$ the BEA provides a minimal degree solution of (10). In order to prove this we need the following lemma:

Lemma 3: Let $(\boldsymbol{c}, \boldsymbol{w})$, $(\boldsymbol{c}', \boldsymbol{w}')$ be polynomial pairs with $ged(\boldsymbol{c})=ged(\boldsymbol{c}')=0$, such that $ged(\boldsymbol{c}'D\boldsymbol{s} - \boldsymbol{w}') > ged(\boldsymbol{c}D\boldsymbol{s} - \boldsymbol{w})$. Then: $deg(\boldsymbol{c}', \boldsymbol{w}') + deg(\boldsymbol{c}, \boldsymbol{w}) \geq ged(\boldsymbol{c}D\boldsymbol{s} - \boldsymbol{w})$.

Corollary: If equality is obtained then $(\boldsymbol{c}', \boldsymbol{w}')$ is a minimal degree solution of (10) for all N in the range $ged(\boldsymbol{c}D\boldsymbol{s} - \boldsymbol{w}) < N \leq ged(\boldsymbol{c}'D\boldsymbol{s} - \boldsymbol{w}')$.

Proof of Lemma 3: The proof is similar to the proof of Lemma 2. It is straight forward to see that

$$ged(\boldsymbol{c}'[\boldsymbol{c}D\boldsymbol{s} - \boldsymbol{w}]) = ged(\boldsymbol{c}D\boldsymbol{s} - \boldsymbol{w}) \cdot \tag{25}$$

Similarly

$$ged(\boldsymbol{c}[\boldsymbol{c}'D\boldsymbol{s} - \boldsymbol{w}']) > ged(\boldsymbol{c}D\boldsymbol{s} - \boldsymbol{w}). \tag{26}$$

Subtracting (25) from (26) we get

$$ged(\boldsymbol{c}\boldsymbol{w}' - \boldsymbol{c}'\boldsymbol{w}) = ged(\boldsymbol{c}D\boldsymbol{s} - \boldsymbol{w}), \tag{27}$$

therefore

$$\begin{aligned} &ged(\boldsymbol{c}D\boldsymbol{s} - \boldsymbol{w}) = ged(\boldsymbol{c}\boldsymbol{w}' - \boldsymbol{c}'\boldsymbol{w}) \\ &\leq deg(\boldsymbol{c}\boldsymbol{w}' - \boldsymbol{c}'\boldsymbol{w}) \leq deg(\boldsymbol{c}, \boldsymbol{w}) + deg(\boldsymbol{c}', \boldsymbol{w}'), \end{aligned} \tag{28}$$

and the proof of Lemma 3 is complete.

Back to Proof of main proposition: The proof will be by induction. Consider the sequence of polynomials $\boldsymbol{g}^{(i)}$, $\boldsymbol{h}^{(i)}$ generated by the BEA relative to the basis polynomials $\mathfrak{G}=\boldsymbol{g}$,$\mathfrak{H}=-1$. Denote the Bezout pair associated with $\boldsymbol{h}^{(i)}$ by $(\boldsymbol{c}^{(i)}, \boldsymbol{w}^{(i)})$. Explicitly:

$$\boldsymbol{h}^{(i)} = \boldsymbol{c}^{(i)}\mathfrak{G} + \boldsymbol{w}^{(i)}\mathfrak{H} = \boldsymbol{c}^{(i)}D\boldsymbol{s} - \boldsymbol{w}^{(i)} \tag{29}$$

The induction hypothesis will contain three assertions:

1. $ged(\boldsymbol{h}^{(i)}) = deg(bp(\boldsymbol{g}^{(i)})) + deg(bp(\boldsymbol{h}^{(i)}))$.
2. $deg(bp(\boldsymbol{g}^{(i)})) = deg(bp(\boldsymbol{h}^{(i+1)}))$.
3. $(\boldsymbol{c}^{(i+1)}, \boldsymbol{w}^{(i+1)})$ is a minimal degree solution of (10) for all N satisfying $ged(\boldsymbol{h}^{(i)}) < N \leq ged(\boldsymbol{h}^{(i+1)})$.

For i=0 all 3 assertions are satisfied.

Suppose the assertion was proved up to i, and we want to prove for i+1.

The pair $(\boldsymbol{c}^{(i+1)}, \boldsymbol{w}^{(i+1)})$ is clearly a solution of (10) for $N = ged(\boldsymbol{h}^{(i+1)})$, but it is not a solution for any larger N, so according to Lemma 3 any pair $(\boldsymbol{c}', \boldsymbol{w}')$ which is a solution for $N > ged(\boldsymbol{h}^{(i+1)})$ must satisfy

$$deg(\boldsymbol{c}', \boldsymbol{w}') + deg(\boldsymbol{c}^{(i+1)}, \boldsymbol{w}^{(i+1)}) \geq ged(\boldsymbol{h}^{(i+1)}). \tag{30}$$

According to the BEA algorithm $\boldsymbol{g}^{(i+1)}$ may be written as

$$\boldsymbol{g}^{(i+1)} = \boldsymbol{h}^{(i+1)} + aD^{\Delta}\boldsymbol{h}^{(i)}, \tag{31}$$

where Δ is set such that

$$ged(\boldsymbol{h}^{(i+1)}) = ged(D^{\Delta}\boldsymbol{h}^{(i)}) = \Delta + ged(\boldsymbol{h}^{(i)}), \tag{32}$$

and a is set to

$$a = -h^{(i+1)}_{ged(\boldsymbol{h}^{(i+1)})} / h^{(i)}_{ged(\boldsymbol{h}^{(i)})}, \tag{33}$$

therefore achieving

$$ged(\boldsymbol{g}^{(i+1)}) > ged(\boldsymbol{h}^{(i+1)}). \tag{34}$$

By definition of the BEA algorithm, (see (21) above), $deg(bp(\boldsymbol{g}^{(i+1)})) > deg(bp(\boldsymbol{h}^{(i+1)}))$, and therefore (31) implies that

$$deg(bp(\boldsymbol{g}^{(i+1)})) = \Delta + deg\left(bp(\boldsymbol{h}^{(i)})\right). \tag{35}$$

According to the first assertion of the induction hypothesis we have $ged(\boldsymbol{h}^{(i)}) = deg(bp(\boldsymbol{g}^{(i)})) + deg(bp(\boldsymbol{h}^{(i)}))$, and according to the second assertion $deg(bp(\boldsymbol{g}^{(i)})) = deg(bp(\boldsymbol{h}^{(i+1)}))$ and therefore

$$\begin{aligned} &ged(\boldsymbol{h}^{(i+1)}) = \Delta + ged(\boldsymbol{h}^{(i)}) = \\ &\left(\Delta + deg(bp(\boldsymbol{h}^{(i)}))\right) + deg(bp(\boldsymbol{g}^{(i)})) = \\ &deg(bp(\boldsymbol{g}^{(i+1)})) + deg(bp(\boldsymbol{h}^{(i+1)})), \end{aligned} \tag{36}$$

so the first assertion is true for i+1. Since $bp(\boldsymbol{g}^{(i+1)})$ satisfies the conditions of $(\boldsymbol{c}', \boldsymbol{w}')$ of (30), the combination of (30) and (36) implies that $bp(\boldsymbol{g}^{(i+1)})$ is a minimal degree solution of (10)

for $N = ged\left(\boldsymbol{h}^{(i+1)}\right)+1$ (at least). But $deg\left(bp(\boldsymbol{h}^{(i+2)})\right) \leq deg\left(bp(\boldsymbol{g}^{(i+1)})\right)$, (see (21) above), and $bp(\boldsymbol{h}^{(i+2)})$ is also a solution of (10) for $N = ged\left(\boldsymbol{h}^{(i+1)}\right)+1$, so the minimality of $bp(\boldsymbol{g}^{(i+1)})$ implies that $deg\left(bp(\boldsymbol{g}^{(i+1)})\right) = deg\left(bp(\boldsymbol{h}^{(i+2)})\right)$ and therefore $\left(\boldsymbol{c}^{(i+2)}, \boldsymbol{w}^{(i+2)}\right) = bp(\boldsymbol{h}^{(i+2)})$ is a minimal solution of (10) for all $ged(\boldsymbol{h}^{(i+1)})<N \leq ged(\boldsymbol{h}^{(i+2)})$ and thus the other 2 assertions are also satisfied for i+1 and the proof is complete.

Actually this BEA process is equivalent to the original BM algorithm as presented in [2]. This is evident by comparing Fig. 1, which is an excerpt from [2], with the Main Proposition and its proof. We leave the full comparison to a more detailed version. We just note that $C(D)$ may be identified with the first coordinate of the Bezout pair of the $\boldsymbol{r}$ polynomials in the BPD process and $B(D)$ may be identified with the first coordinate of the Bezout pair of $\boldsymbol{h}^{(i)}$ while the second coordinate is ignored in [2]. Also $\boldsymbol{g}^{(i)}, \boldsymbol{h}^{(i)}, \boldsymbol{r}$ do not appear explicitly in [2], but they are present implicitly via the computation of d. L of [2] is the degree of the Bezout pair, (which does not always coincide with the degree of $C(D)$). Equivalence between the riBM algorithm of [3] and our approach is even simpler, but due to space limitations a comparison is not included in this version.

1) $1 \rightarrow C(D)$ $\quad 1 \rightarrow B(D)$ $\quad 1 \rightarrow x$
   $0 \rightarrow L$ $\quad 1 \rightarrow b$ $\quad 0 \rightarrow N$
2) If $N = n$, stop. Otherwise compute

$$d = s_N + \sum_{i=1}^{L} c_i s_{N-i}.$$

3) If $d = 0$, then $x + 1 \rightarrow x$, and go to 6).
4) If $d \neq 0$ and $2L > N$, then
   $C(D) - d\,b^{-1} D^x B(D) \rightarrow C(D)$
   $x + 1 \rightarrow x$
   and go to 6).
5) If $d \neq 0$ and $2L \leqq N$, then
   $C(D) \rightarrow T(D)$ [temporary storage of $C(D)$]
   $C(D) - d\,b^{-1} D^x B(D) \rightarrow C(D)$
   $N + 1 - L \rightarrow L$
   $T(D) \rightarrow B(D)$
   $d \rightarrow b$
   $1 \rightarrow x$.
6) $N + 1 \rightarrow N$ and return to 2).

Fig. 1. Original Massey Formulation of BM Algorithm

X. Fast BCH Decoding

In the context of BCH codes it is well known that $s_{2N} = s_N^2$ for all $N \geq 1$. It follows immediately that $(1+D\boldsymbol{s})^2 \bmod D^{2t+1}$ is a polynomial containing the even powers of $1+D\boldsymbol{s}$ and $(1+D\boldsymbol{s})D\boldsymbol{s} \bmod D^{2t+1}$ contains the odd powers of $1+D\boldsymbol{s}$. Setting $\boldsymbol{g}=D\boldsymbol{s}$, $\boldsymbol{h}=1+D\boldsymbol{s}$, (as in the previous sections), we see that for every polynomial pair $(\boldsymbol{a}, \boldsymbol{w})$ we have:

$$ged(\boldsymbol{ag} + \boldsymbol{wh}) = ged(\boldsymbol{ahg} + \boldsymbol{wh}^2). \tag{37}$$

The first step of a BEA applied to $\boldsymbol{hg}, \boldsymbol{h}^2$ (with Bezout pairs computed relative to $\boldsymbol{hg}, \boldsymbol{h}^2$) will be to multiply $\boldsymbol{h}^2$ by a scalar multiple of an odd power of D in order to align it with $\boldsymbol{hg}$. From that point on all the polynomials involved in the process are odd power polynomials. Therefore in each iteration the ged increases by an even number, and (37) implies that this is true for the BEA applied to $\boldsymbol{g}, \boldsymbol{h}$ as well. After at most t iterations we have $ged(\boldsymbol{ahg} + \boldsymbol{wh}^2) \geq 2t+1$, and therefore also $ged(\boldsymbol{ag} + \boldsymbol{wh}) \geq 2t+1$. Setting $\boldsymbol{c} = \boldsymbol{a}+\boldsymbol{w}$ we see that the pair $(\boldsymbol{c}, \boldsymbol{w})$ is a minimal degree solution of (10) for $N = 2t + 1$. So we see that for BCH codes the BEA, (and the equivalent BM algorithm), converges to the solution of the key equation in $\leq t$ iterations. A different proof of this appears in [1], but our proof based on the BEA is much simpler. Moreover it shows that not only the number of iterations is reduced, but also the sizes of the polynomials since the BEA may be applied to $\boldsymbol{hg} \bmod D^{2t+1}$ and $\boldsymbol{h^2} \bmod D^{2t+1}$ which contain only half the coefficients of $\boldsymbol{g}$ and $\boldsymbol{h}$.

XI. Erasure Decoding

Let $\boldsymbol{s} = \boldsymbol{Ha}$ be a syndrome of a received word $\boldsymbol{a}$. Suppose $\boldsymbol{a}$ contains $2u \leq 2t$ erasures. Let $\boldsymbol{u}$ denote the locator polynomial of the erasures so $deg(\boldsymbol{u}) = 2u$. The polynomial $\boldsymbol{u}$ may be written as $\boldsymbol{u} = \boldsymbol{c_u q}$, where $\boldsymbol{c_u}$ is the locator polynomial of the errors in $\boldsymbol{u}$, and $\boldsymbol{q}$ is the locator polynomial of the non-errors in $\boldsymbol{u}$. The ELP of $\boldsymbol{s}$ may be denoted as $\boldsymbol{c} = \boldsymbol{c_o c_u}$, wherein $\boldsymbol{c_o}$ is the locator polynomial of the errors outside $\boldsymbol{u}$. Therefore there exist $\boldsymbol{w}$ with $deg(\boldsymbol{w}) \leq \deg(\boldsymbol{c})$ such that $ged(\boldsymbol{cDs} - \boldsymbol{w}) \geq 2t+1$. Due to space limitations we leave to the reader to show that if $deg(\boldsymbol{c_o}) \leq t\text{-}u$ then $(\boldsymbol{c_o}, \widetilde{\boldsymbol{w}})$ is the unique minimal solution with $deg\ (\boldsymbol{c_o}, \widetilde{\boldsymbol{w}}) \leq t - u$ of the equation

$$\boldsymbol{x}(D\tilde{\boldsymbol{s}}) - \boldsymbol{y} = 0 \bmod D^{2(t-u)+1}, \tag{38}$$

where

$$\begin{aligned} \tilde{\boldsymbol{s}} &= (\boldsymbol{us} - \boldsymbol{us} \bmod D^{2u})/D^{2u} \\ \widetilde{\boldsymbol{w}} &= -[\boldsymbol{c_o}(\boldsymbol{u}D\boldsymbol{s} \bmod D^{2u+1}) - \boldsymbol{qw}]/D^{2u}. \end{aligned} \tag{39}$$

Computation of the polynomials $\boldsymbol{q}, \boldsymbol{c_u}, \boldsymbol{w}$ follows immediately.

References

[1] E.R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, chapter 7 (1968).

[2] J.L. Massey, "Shift register synthesis and BCH decoding", IEEE Trans. Inform. Theory, vol. IT-15, pp. 122–127 (1969).

[3] D. V. Sarwate and N. R. Shanbhag, "High-speed architectures for Reed-Solomon decoders," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 9, no. 5, pp. 641-655, Oct. 2001.

[4] F.J. MacWilliams, N.J.A. Sloane, The Theory of Error-Correcting Codes, Volume 16, North-Holland Mathematical Library.

[5] K. Imamura and W. Yoshida, "A simple derivation of the Berlekamp-Massey algorithm and some applications", in IEEE Transactions on Information Theory, vol. 33, no. 1, pp. 146-150, Jan 1987.

[6] M. Bras-Amoros M. E. O'Sullivan, "The Berlekamp-Massey Algorithm and the Euclidean Algorithm: a Closer Link", Available: https://arxiv.org/abs/0908.2198

[7] J.L. Dornstetter. "On the equivalence between Berlekamp's and Euclid's Algorithms", IEEE Trans. Inform. Theory, 33(3):428–431, 1987.

[8] A. E. Heydtmann and J. M. Jensen. "On the equivalence of the Berlekamp-Massey and the Euclidean algorithms for decoding", IEEE Trans. Inform. Theory, 46(7):2614–2624, 2000.

[9] T. D. Mateer, "On the equivalence of the Berlekamp-Massey and the euclidean algorithms for algebraic decoding", 12th Canadian Workshop on Information Theory, Kelowna, BC, 2011, pp. 139-142.

[10] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. "A method for solving key equation for decoding Goppa codes", Information and Control, 27:87–99, 1975.