ENCODING AND DECODING OF BCH CODES USING LIGHT AND SHORT CODEWORDS

RON M. ROTH [*]

AND

GADIEL SEROUSSI [**]

ABSTRACT

It is shown that every q-ary primitive BCH code of designed distance δ and sufficiently large length n contains a codeword $\mathbf{c}_0$ of weight $w = O(\delta)$, and degree $\deg(\mathbf{c}_0) = o(n)$. These so called *light and short* codewords are used to describe encoding and decoding algorithms which run on sequential machines in time $O(\delta n)$, i.e. linear in n for fixed δ. For high-rate primitive BCH codes this is faster than the commonly used algorithms, which are nonlinear in n when run on sequential machines.

---

[*] Department of Computer Science, Technion, Israel Institute of Technology, Haifa 32000 - Israel.

[**] Currently at Cyclotomics, 2120 Haste St., Berkeley, CA 94704, on leave from the Technion, Israel Institute of Technology.

Part of this work was presented at the IEEE International Symposium on Information Theory, Ann Arbor, Michigan, October 1986.

I. INTRODUCTION

Let C be a BCH code of length n and designed distance δ over $GF(q)$. As it is well known [2, chs. 5,7][6, chs. 7-9], C is a cyclic code generated by the polynomial $g(x)$ of least degree such that $0 = g(\alpha^b) = g(\alpha^{b+1}) = \cdots = g(\alpha^{b+\delta-2})$, where b is an arbitrary integer, and α is a primitive n-th root of unity, found in the splitting field $GF(q^m)$ of $x^n - 1$. Usually, $b = 1$, and C is called a *narrow-sense* BCH code. The minimum distance d of C satisfies the BCH bound

$$d \geq \delta \,. \tag{1}$$

C is called a *primitive* BCH code if $n = q^m - 1$.

The degree of $g(x)$ is in general bounded from above by

$$\deg\,(g(x)) \leq (\delta - 1)m.$$

In the case of binary, narrow sense BCH codes, this bound can be tightened to

$$\deg\,(g(x)) \leq \frac{(\delta - 1)}{2}\, m.$$

An extensive discussion on the determination of the exact degree of $g(x)$ can be found in [2, ch. 12].

Let $\mathbf{c}$ be a codeword in a BCH code C. We denote by $\mathrm{wt}\,(\mathbf{c})$ the Hamming weight (number of nonzero entries) of $\mathbf{c}$, and by $\deg\,(\mathbf{c})$ the degree of the polynomial $c(x) \in GF(q)[x]/(x^n - 1)$ associated with $\mathbf{c}$. Also, we use the standard asymptotic notation $O(\delta)$ for a function $f(\delta)$ bounded above by $\lambda\delta$ for some constant λ, and $o(n)$ for a function $h(n)$ such that $\lim_{n\to\infty} h(n)/n = 0$. For the sake of simplicity, we shall use the $O(\,)$ and $o(\,)$ symbols in somewhat loose way, mentioning sometimes "a codeword" $\mathbf{c} \in C$, of weight $O(\delta)$ and degree $o(n)$. Actually, we are referring to an infinite sequence of BCH codes $(C)_{n,\delta}$ and an infinite sequence of codewords $(\mathbf{c})_{n,\delta}$, parametrized by n and δ, with $(\mathbf{c})_{n,\delta} \in (C)_{n,\delta}$, $\mathrm{wt}\,((\mathbf{c})_{n,\delta}) = O(\delta)$, and $\deg\,((\mathbf{c})_{n,\delta}) = o(n)$. This precise meaning is implicitly assumed in the sequel for all asymptotic assertions about codewords.

This paper deals with primitive (not necessarily narrow-sense) BCH codes. We show that for a sufficiently long primitive BCH code C of designed distance δ, there exists a nonzero codeword $\mathbf{c}_0 \in C$, such that $\mathrm{wt}\,(\mathbf{c}_0) = O(\delta)$, and $\deg\,(\mathbf{c}_0) = o(n)$. More specifically, we prove the following results:

Theorem 1. *Let C be a primitive BCH code of length n and designed distance δ over $GF(q)$. Let s and ω be positive integers satisfying*

$$\frac{\omega}{q-1}(n+1)^{\frac{\delta-1}{\omega}} < s \le n. \tag{2}$$

Then, there exists a nonzero codeword $\mathbf{c}_0 \in C$ such that

$$\mathrm{wt}(\mathbf{c}_0) \le 2\omega,$$

and

$$\deg(\mathbf{c}_0) < s.$$

Corollary 1. *Let C be a code as in Theorem 1, with n and δ satisfying*

$$\delta < 1 + \frac{(q-1)n}{2\sqrt{n+1}}. \tag{3}$$

Then, there exists a nonzero codeword $\mathbf{c}_0 \in C$ such that

$$\mathrm{wt}(\mathbf{c}_0) \le 4(\delta-1), \tag{4}$$

and

$$\deg(\mathbf{c}_0) \le 2\frac{\delta-1}{q-1}\sqrt{n+1}. \tag{5}$$

We say that a codeword $\mathbf{c}$ is *short* if $\deg(\mathbf{c}) = o(n)$, and that $\mathbf{c}$ is *light* if $\mathrm{wt}(\mathbf{c}) = O(\delta)$. Note that if $\lim_{n\to\infty} \delta / \sqrt{n} = 0$, then (4)-(5) imply that $\mathbf{c}_0$ is both light and short.

The proofs of Theorem 1 and Corollary 1 are presented in Section II, where we also show that these results can be strengthened in the special (and very common) case of $q = 2$ and $b = 1$, i.e. binary narrow-sense BCH codes.

In Sections III and IV we use the light and short codewords mentioned above to describe procedures for encoding and decoding primitive BCH codes. For $\delta \le O(\sqrt{n}/\log n)$, the proposed encoding procedure

runs in time $O(\delta n)$, which is *linear* in n for fixed δ. The technique used is similar to one proposed by Kaminski [3] for finding the remainder of a long polynomial (or integer) divided by a short one.

Notice that the sequential complexity of the standard encoding procedure for BCH (and other cyclic) codes [2, ch. 5][6, ch. 7] is $O(\delta n \log n)$, that is, nonlinear in n. In hardware implementations, this nonlinearity is usually concealed by using a shift register architecture, which is in essence a parallel machine. Encoding is done in n steps, with $O(\delta \log n)$ operations being performed in parallel at each step. When this procedure is implemented in a sequential environment (e.g. a computer program), however, its complexity becomes $O(\delta n \log n)$.

On the decoder side, we show that if $\delta \le O(\sqrt{n/\log^3 n})$ then the syndrome computation dominates the running time of the standard decoding procedure for BCH codes [2, ch. 7][6, ch. 9]. Since computing the syndrome is computationally equivalent to encoding, our complexity upper bound of $O(\delta n)$ applies also for decoding.

We conclude that for high-rate primitive BCH codes, the proposed encoding and decoding procedures are faster than the standard ones when implemented on sequential machines. Some numerical examples of light and short codewords, and of encoding running times are presented in the concluding Section V, and in the Appendix.

II. EXISTENCE OF LIGHT AND SHORT CODEWORDS

The following lemma is a somewhat stronger version of Theorem 1:

Lemma 1. *Let C be a primitive BCH code of length n and designed distance δ over $GF(q)$. Let s and ω be positive integers satisfying $s \le n$ and*

$$\sum_{l=0}^{\omega} \binom{s}{l} (q-1)^l > (n+1)^{\delta-1} . \tag{6}$$

Then, there exists a nonzero codeword $\mathbf{c}_0 \in C$ such that

$$\mathrm{wt}\,(\mathbf{c}_0) \le 2\omega\,,$$

and

$$\deg\,(\mathbf{c}_0) < s\,.$$

Proof. Consider the set

$$V(\omega, s) = \{\, \mathbf{v} \in GF(q)^s \mid \mathrm{wt}\,(\mathbf{v}) \le \omega \,\}.$$

The cardinality of $V(\omega, s)$ is given by the sum at the left hand side of (6). For each $\mathbf{v} \in V(\omega, s)$ we consider the syndrome vector

$$[\, v(\alpha^b), v(\alpha^{b+1}), \cdots, v(\alpha^{b+\delta-2})] \in GF(q^m)^{\delta-1}\,,$$

where $v(x)$ is the polynomial of degree (at most) $s-1$ associated with $\mathbf{v}$. Recalling that $n+1 = q^m$, we notice that the number of possible distinct syndrome vectors is $(n+1)^{\delta-1}$, which, by (6), is less than the cardinality of $V(\omega, s)$. It follows that there must exist two distinct vectors $\mathbf{v}_1, \mathbf{v}_2 \in V(\omega, s)$ with the same syndrome vector. Hence, $\mathbf{v}_1 - \mathbf{v}_2$ (suitably extended with zeros to length n) is a codeword of C with the desired properties. □

Lemma 2. *For every two integers* $s \ge \omega > 0$,

$$\binom{s}{\omega} \ge \left(\frac{s}{\omega}\right)^{\omega}.$$

Proof. The claim follows immediately from the definition of the binomial coefficient $\binom{s}{\omega}$, and from the fact that for any $a \ge b > 1$, we have

$$\frac{a-1}{b-1} \ge \frac{a}{b}\,. \qquad \square$$

Proof of Theorem 1. It can be readily verified that the condition (2) of the theorem implies

$$\left(\frac{s}{\omega}\right)^{\omega} (q-1)^{\omega} > (n+1)^{\delta-1}. \tag{7}$$

Together with the result of Lemma 2, (7) implies the condition (6) of Lemma 1. The latter establishes the

existence of a codeword $\mathbf{c}_0$ with the desired properties. □

Proof of Corollary 1. Let $\omega = 2(\delta - 1)$, and let

$$s = 1 + \left\lfloor 2\,\frac{\delta - 1}{q - 1}\sqrt{n+1} \right\rfloor.$$

It follows from these definitions that

$$s > \frac{\omega}{q-1}(n+1)^{\frac{\delta-1}{\omega}} = 2\,\frac{\delta-1}{q-1}\sqrt{n+1}\,.$$

Also, by (3), $s \le n$. The claim of the corollary now follows by direct application of Theorem 1. □

The bound (4) on the weight of the light and short codeword $\mathbf{c}_0$ of Corollary 1 can be strengthened at the expense of weakening the bound (5) on the degree. The result is presented in the following corollary.

Corollary 2. *Let C be a code as in Theorem 1, with n and δ satisfying*

$$n > \left(1 + \frac{1}{n}\right)^{\delta-1}\left(\frac{\delta}{q-1}\right)^{\delta}.$$

Then, there exists a nonzero codeword $\mathbf{c}_0 \in C$ such that

$$\mathrm{wt}\,(\mathbf{c}_0) \le 2\delta\,,$$

and

$$\deg(\mathbf{c}_0) \le \frac{\delta}{q-1}(n+1)^{\frac{\delta-1}{\delta}}.$$

Proof. Let $\omega = \delta$. The proof then follows along the same lines as that of Corollary 1. □

We consider now the special case $q = 2$ and $b = 1$, i.e., the case of *binary narrow-sense primitive* BCH (in short, BNP-BCH) codes. We assume that δ is odd, i.e., $\delta = 2t + 1$, t being the *designed correction capability* of the code.

The following is a restatement of Theorem 1, applied to the BNP case:

Theorem 2. *Let C be a BNP-BCH code of length n and designed distance $\delta = 2t+1$. Let s and ω be positive integers satisfying*

$$\omega(n+1)^{\frac{t}{\omega}} < s \le n .$$

Then, there exists a nonzero codeword $\mathbf{c}_0 \in C$ such that $\mathrm{wt}(\mathbf{c}_0) \le 2\omega$ *and* $\deg(\mathbf{c}_0) < s$.

Proof. The proof follows along the same lines as that of Lemma 1 and Theorem 1, except for the fact that here it suffices to evaluate the syndrome elements only at the odd powers of α, i.e., at $\alpha, \alpha^3, \cdots, \alpha^{2t-1}$. □

The following corollary is obtained by setting $\omega = 2t$ and $s = 1 + \left\lfloor 2t\sqrt{n+1} \right\rfloor$ in Theorem 2.

Corollary 3. *Let C be a code as in Theorem 2, with n and t satisfying $t < \dfrac{n}{2\sqrt{n+1}}$. Then, there exists a nonzero codeword $\mathbf{c}_0 \in C$ such that* $\mathrm{wt}(\mathbf{c}_0) \le 4t$ *and* $\deg(\mathbf{c}_0) \le 2t\sqrt{n+1}$.

An interesting example of the application of Corollary 3 is obtained by substituting $t = 1$. The resulting code is a single-error-correcting binary Hamming code which is generated by a binary primitive polynomial of degree $m = \log_2(n+1)$. Corollary 3 implies that the code contains a codeword of weight at most 4 and degree at most $2\sqrt{n+1}$.

III. USING LIGHT AND SHORT CODEWORDS FOR ENCODING

Let C be an $[n, k]$ primitive BCH code over $GF(q)$, with generator polynomial $g(x)$. The most commonly used procedure for encoding C can be described as follows: let $(u_0, u_1, \cdots, u_{k-1})$ be the message vector, and let $u(x)$ be its associated polynomial. Compute the remainder $r(x) = x^{n-k}u(x) \mod g(x)$. The transmitted codeword is then $c(x) = x^{n-k}u(x) - r(x)$.

In the sequel we assume that $\delta \le O(\sqrt{n} / \log n)$, that is, we consider only high-rate primitive BCH codes. One way of obtaining the residue $r(x)$ is to carry out a *long division* [4] of $x^{n-k}u(x)$ by $g(x)$. We refer to this as the *naive* way. The number of arithmetic operations over $GF(q)$ required in the naive procedure is $O(n \deg(g))$, which for high-rate BCH codes is $O(n\,\delta \log n)$ [6, p. 263]. Hence, this division method, although easy to implement on sequential machines, yields a running time nonlinear in n. Notice that the widely used shift-register encoder [2, ch. 5][6, ch. 7] is in fact a variation of long division, and has a similar arithmetic complexity. A more sophisticated division method is given by Schönhage's algorithm ([10], [1, §8.3]), whose complexity is $O(n \cdot \log \deg(g) \cdot \log\log \deg(g))$. This algorithm is asymptotically faster than the long division algorithm. However, it is quite complicated to implement, and it is still not linear in n for fixed δ.

We present now an encoding procedure which uses the light and short codewords whose existence was established in Section II. Let $\mathbf{c}_0$ denote such a codeword. The encoding procedure consists of the following two steps:

(a) Compute (naively) $r_0(x) = x^{n-k}u(x) \bmod c_0(x)$.

(b) Compute (naively) $r(x) = r_0(x) \bmod g(x)$.

Since $c_0(x)$ is a multiple of $g(x)$, $r(x)$ is the desired remainder. Step (a) has complexity $O(\text{ wt}(\mathbf{c}_0)\, n) = O(\delta n)$, and Step (b) has complexity $O(\deg(\mathbf{c}_0) \deg(g)) = O(\delta^2 \sqrt{n} \log n)$. Since $\delta \le O(\sqrt{n} / \log n)$, Step (b) is at least as fast as Step (a), resulting in an overall encoding complexity of $O(\delta n)$.

In describing the above procedure, we assume that a specific light and short codeword $\mathbf{c}_0$ is known. Such a codeword can be searched for exhaustively by generating all vectors in $V(4(\delta-1), s)$ (as defined in the proof of Lemma 1) for $s = O(\delta\sqrt{n+1})$ until a codeword is found (Corollary 1 guarantees that one will be found). The complexity of the search is roughly $O(s^{4(\delta-1)})$, which is polynomial in n for fixed δ. Although this might still be a heavy computation, it should be noted that the light and short codeword $\mathbf{c}_0$ needs to be searched for only once for a given code. A list of light and short codewords for some binary

BCH codes, found in the above manner, is given in the Appendix.

IV. USING LIGHT AND SHORT CODEWORDS FOR DECODING

We refer now to the standard decoding procedure of BCH codes using the Berlekamp-Massey algorithm ([2, §7.4], [7]).

Given a received word $\mathbf{y} = \mathbf{c} + \mathbf{e}$ ($\mathbf{e}$ being the error word), the decoding procedure consists of the following steps:

(i) Evaluate the syndrome vector

$$S_i = y(\alpha^{b+i}), \qquad 0 \le i \le \delta - 2.$$

This step can be carried out by first finding the remainder $y(x)$ mod $g(x)$ using the technique described in Section III, and then applying a $(\delta-1)\log_q(n+1) \times (\delta-1)\log_q(n+1)$ linear transformation to obtain the syndrome values. The overall complexity of this step is, therefore, $O(\delta n)$.

(ii) Find the Error-Locator Polynomial (ELP) as the linear recursion of $S_0, S_1, \cdots, S_{\delta-2}$ using the Berlekamp-Massey algorithm.

This step requires $O(\delta^2)$ operations over $GF(q^m)$. Translating naively the arithmetic of $GF(q^m)$ to that of $GF(q)$ results in a complexity of $O(\delta^2 \log^2 n)$.

(iii) Find the roots of the ELP in $GF(q^m)$.

This can be done using Rabin's algorithm [9] $\frac{\delta-1}{2}$ times. The arithmetic complexity is $O(\delta \log n\, \delta^2)$ operations over $GF(q^m)$, which result in $O(\delta^3 \log^3 n)$ operations over $GF(q)$.

(iv) Find error locations by extracting the logarithms of the roots of the ELP.

A simple known algorithm for logarithm extraction [5, p. 9] takes $O(\sqrt{n} \log^2 n)$ $GF(q)$-operations for each

root. Therefore, the overall complexity of this step is $O(\delta\sqrt{n}\log^2 n)$.

(v) Find the error values (unnecessary for the binary case).

The complexity is similar to that of Step (ii) [6, p. 246].

(vi) Correct $y(x)$.

This is done in $O(\delta)$ operations.

Summing up the complexities of the different steps, it can be readily verified that Step (i) is dominant if $\delta \le O(\sqrt{n/\log^3 n})$. Given this condition (which is somewhat stronger than the one required for the encoding algorithm), the overall complexity of the decoding procedure is $O(\delta n)$.

V. CONCLUSION

The existence of light and short codewords in high-rate primitive BCH codes leads to an encoding-decoding algorithm whose complexity is *linear* in the code length, provided that the designed distance is fixed. This performance is achieved by implementing a modular polynomial division in two steps, one of which utilizes the light and short codeword. The complexity of other known encoding-decoding algorithms is not smaller than $O(n \cdot \log\log n \cdot \log\log\log n)$ in sequential environments. Hence, using light and short codewords in the encoding-decoding scheme yields a better asymptotic running time for high-rate BCH codes.

We discuss now some practical considerations concerning the design of an encoder [decoder] for a *specific* BNP-BCH code C of length n and designed distance $\delta = 2t + 1$. Let $\mathbf{c}_0$ be a light and short codeword of C with $\deg(\mathbf{c}_0) = s_0$ and $\mathrm{wt}(\mathbf{c}_0) = w_0$. Also, let s_g and w_g be the degree and the weight of the generator polynomial $g(x)$, respectively. Then, the number N_x of XOR (exclusive-or) operations required to perform the encoding process, as given in Section III, is given by

$$N_x = (n - s_0) \cdot w_0 + (s_0 - s_g) \cdot w_g \,. \tag{8}$$

Clearly, if $\mathbf{c}_0 = \mathbf{g}$, then $N_x = (n - s_g) \cdot w_g$, which is the number of XOR operations in the naive long division algorithm. The goal of the designer is to minimize the right-hand side of (8) under the constraint that both s_0 and w_0 correspond to some codeword of C. In particular, $s_0 \geq f(w_0)$, $f(w)$ being the lowest degree of any nonzero codeword in C having weight $\leq w$. Unfortunately, the codeword $\mathbf{c}_0$ which optimizes (8) is generally hard to find. The same applies to the values of the function $f(\cdot)$, although a simple upper bound for the latter can be derived by the same arguments which lead to Lemma 1: it is easy to verify that for each w,

$$\sum_{l=0}^{\left\lfloor \frac{w}{2} \right\rfloor} \binom{f(w)}{l} \leq (n+1)^t \,.$$

Nevertheless, one may get a significant improvement in the complexity of a specific code just by picking one light and short codeword (which can be found in an exhaustive procedure as mentioned in Section III) and checking the achieved gains in complexity. As an illustrative example, consider a [255, 231, 7] binary BCH code. The generator polynomial $g(x)$ and a light and short codeword $\mathbf{c}_0$ for this code are presented in the Appendix (code No. 2). Encoding this code using a naive long division procedure would require 3,465 XOR operations, compared with 1,904 ones if $\mathbf{c}_0$ is used. A similar ratio is achieved in code No. 3. Note that sometimes the generator polynomial itself can be regarded as a light and short codeword (this is the case for the [127, 106, 7] code, No. 1 in the Appendix).

Finally, we would like to point out that Theorem 1, as well as the other results of Sections I and II, can be generalized by taking *any* set of s coordinates, not necessarily the low-order ones. For example, Corollary 1 can be reformulated as follows: For every code C as in Theorem 1 which satisfies (3), and for *any* set A of $1 + 2\frac{\delta - 1}{q - 1}\sqrt{n+1}$ coordinates, there exists a nonzero codeword $\mathbf{c}_0$ with $\mathrm{wt}(\mathbf{c}_0) \leq 4(\delta - 1)$, whose support is contained in A.

APPENDIX: LIGHT AND SHORT CODEWORDS OF SOME BINARY BCH CODES

The following is a list of examples of light and short codewords of some common binary narrow-sense BCH codes. In the following, $M_1(x)$ denotes the minimal polynomial of α and, in a similar manner, $M_i(x)$ denotes the minimal polynomial of α^i. Each polynomial is described by its support, e.g., the polynomial $\{ 0,1,7 \}$ stands for $x^7 + x + 1$. Recall also that

$$g(x) = \prod_{j=1}^{t} M_{2j-1}(x) .$$

A list of minimal polynomial can be found in [8, Appendix C], which also assisted in finding the following generator polynomials.

1. $n = 127, \quad \delta = 7 \ (t = 3), \quad M_1(x) = \{ 0,1,7 \}$,

$g(x) = \{ 0,1,3,4,12,13,16,19,21 \}$,
$c_0(x) = \{ 0,1,3,8,9,14,21,31 \}$.

2. $n = 255, \quad \delta = 7 \ (t = 3), \quad M_1(x) = \{ 0,2,3,4,8 \}$,

$g(x) = \{ 0,2,4,5,7,8,13,15,16,17,19,20,21,23,24 \}$,
$c_0(x) = \{ 0,3,11,13,16,20,25,32 \}$.

3. $n = 511, \quad \delta = 7 \ (t = 3), \quad M_1(x) = \{ 0,4,9 \}$,

$g(x) = \{ 0,3,4,5,6,8,9,11,13,16,21,22,24,26,27 \}$,
$c_0(x) = \{ 0,18,39,43,53,68,71,73 \}$.

4. $n = 1023, \quad \delta = 7 \ (t = 3), \quad M_1(x) = \{ 0,3,9 \}$,

$g(x) = \{ 0,1,4,8,12,16,19,21,23,28,30 \}$,
$c_0(x) = \{ 0,15,59,69,79,112,122,124 \}$.

5. $n = 127, \quad \delta = 9 \ (t = 4), \quad M_1(x) = \{ 0,1,7 \}$,

$g(x) = \{\ 0, 1, 2, 3, 4, 5, 8, 11, 12, 13, 14, 16, 18, 21, 23, 27, 28\ \}$,
$c_0(x) = \{\ 0, 18, 25, 27, 30, 34, 37, 38, 40, 53\ \}$.

6. $n = 255$, $\delta = 8$ ($t = 4$), $M_1(x) = \{\ 0, 2, 3, 4, 8\ \}$,

$g(x) = \{\ 0, 2, 3, 4, 5, 6, 7, 9, 14, 16, 17, 19, 20, 22, 25, 26, 27, 29, 30, 31, 32\ \}$,
$c_0(x) = \{\ 0, 2, 9, 13, 14, 17, 21, 30, 33, 34, 35, 37, 43, 44\ \}$.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA, 1974.

[2] E. R. Berlekamp, *Algebraic Coding Theory.* Revised Edition, Aegean Park Press, Laguna Hills, CA, 1984.

[3] M. Kaminski, "Algorithms for integer and polynomial multiplication", Ph.D. Thesis, The Hebrew University of Jerusalem, 1982 (in Hebrew).

[4] D. E. Knuth, *The Art of Computer Programming, vol. 2: Seminumerical Algorithms.* Second edition, Addison-Wesley, Reading, MA, 1981.

[5] D. E. Knuth, *The Art of Computer Programming, vol. 3: Sorting and Searching.* Addison-Wesley, Reading, MA, 1973.

[6] F. J. MacWilliams, N. J. A. Sloane, *The Theory of Error-Correcting Codes.* North-Holland, Amsterdam, 1977.

[7] J. L. Massey, "Shift register synthesis and BCH decoding", *IEEE Transactions on Information Theory,* vol. IT-15, 1969, pp. 122-127.

[8] W. W. Peterson, E. J. Weldon, *Error-Correcting Codes.* Second edition, MIT Press, Cambridge, MA, 1972.

[9] M. O. Rabin, "Probabilistic algorithms in finite fields", *SIAM J. Comp.*, vol. 9, No. 2, May 1980, pp. 273-280.

[10] A. Schönhage, "Schnelle multiplikation von polynomen über körpern der charakteristic 2", *Acta Informatica*, 7 (1977), pp. 395-398.