MSRI-UP 2009 – SEMINAR IN CODING THEORY – DISCUSSIONS

1. General Information and Goals

The discussion meeting each day of the seminar will be devoted to preliminary work on an assigned list of problems in groups of three. I expect that each group will need to continue work on the assignment during the afternoon and the evening. The solutions will be due at the next meeting of the seminar.

The goals of this part of MSRI-UP are to practice using the concepts we have discussed in the lectures, to work through many examples, and to practice working on mathematics *in groups like the research groups later in the program.*

A few words about this way of working are probably in order. In these meetings, we will all be aiming for *collaborative learning* – that is for an integrated group effor in analyzing and attacking the discussion questions. The ideal is for everyone in each of the groups to be fully involved in the process. The idea is that, by actively participating by talking about the ideas yourself in your own words, you can come to a better first understanding of what is going on than if you simply listen to someone else talk about those ideas.

However, it must be said that to get the most out of this approach, you may need to adjust some of your thinking. In particular,

- This is *not a competition* in any sense. You and your fellow group members are working as a team, and the idea is to have everyone understand what the group does fully.
- At different times, it is inevitable that different people in the group will have a more complete grasp of the questions at hand and others will have a less complete grasp. Dealing with this in a group setting is excellent preparation for real work in a team; it also offers opportunities for significant educational experiences.
- If you feel "totally clueless" at some point, your role will be to *ask questions* and, if necessary, pester the others in your group until the point has been worked out to your full satisfaction. (Don't forget–the others in the group may be jumping to unwarranted conclusions and your questions may save the group from pursuing an erroneous train of thought.)
- On the other hand, when you think you really see how something works, you may need to explain it carefully to others. (Don't forget–the absolutely best way to make sure you really understand something is to try to explain it to someone else(!) If you are skipping over an important point in your thinking, it can become very apparent when you set out to explain your thinking to the other members of your team.)

In short, everyone has important things to contribute, and everyone will contribute in different ways at different times.

Date: May 13, 2009.

Week 1 – Beginnings of Coding Theory

Day 1 – Basic Notions. We have introduced the basic language of binary block codes (subsets of $\mathbb{F}_2^n$, so the codewords consist of strings of some fixed number n of 0s and 1s).

A) Recall that the *Hamming distance* $d(v, w)$ is the number of positions in which the words differ. Show that the Hamming distance has the usual properties of a *metric, or distance function*:

1) For all v, w, $d(v, w) \geq 0$ and $d(v, w) = 0$ if and only if $v = w$.
2) For all v, w, $d(v, w) = d(w, v)$.
3) For all u, v, w, $d(u, w) \leq d(u, v) + d(v, w)$ (a triangle inequality).

B) Determine the *minimum distance* of each of the following codes:

1) $C = \{1010, 1100, 0011\}$
2) $C = \{000000, 111000, 000111, 111111\}$
3) $C =$ the code obtained from $\mathbb{F}_2^n$ by adding a parity check digit. (That is, if $w \in K^n$ we have $w0$ in C if w has an even number of 1's and $w1$ if w has an odd number of 1's.)

C) Suppose a binary symmetric channel with reliability $p = 1 - 10^{-3} = .99$ is used to communicate words of length 11 bits at a rate of 10^5 bits per second.

1) With no error control at all, how many words per second would we expect to be transmitted incorrectly? Explain.
2) Now suppose that a parity check digit is added to all words (as in question B part 3), giving codewords of length $n = 12$, so that a single bit error (or any odd number of bit errors can be *detected* if they occur. How many words per second would we expect to be transmitted incorrectly but not detected? Indicate whether your answer is exact or an approximation, and explain?
3) We will see later in the seminar that it is possible to encode words of length 11 to codewords of length 15 (by adding 4 additional parity check digits) in such a way that the resulting code has minimum distance 3. Hence a nearest-codeword decoder can correct any single bit error in a received word. However, if there are two or more bit errors in the received word, then the decoding will be *incorrect.* Assuming we have a decoder (we will see how to construct one), how many words per second would we expect to be decoded incorrectly?
4) Discuss the tradeoffs in efficiency and reliability of communication in the scenarios from parts 1,2,3.

D) Error control in the form of check digits is often used for information using nonbinary alphabets. A case you may have run into is the ISBN number used by publishers to identify books. For instance, the ISBN number of the HHLLPRW text we are using is 0824704657 (see the barcode on the back cover). All ISBN's have the form $d_1 d_2 \cdots d_9; d_{10}$ where the first 9 digits are ordinary decimal digits 0–9 and are the actual identifier for the book. The last digit d_{10} is a check digit that is added to make the number

$$d_1 + 2d_2 + 3d_3 + \cdots + 9d_9 + 10d_{10} \tag{1.1}$$

evenly divisible by 11. (In cases where d_{10} would need to be 10 to make this work, an X is used instead of a digit.)

1) Check that (1.1) is divisible by 11 for the ISBN number from our book.
2) Explain why given any $d_1, \ldots, d_9$, there is *always* an integer $d_{10} \in \{1, 2, \ldots, 10\}$ such that (1.1) holds.
3) Show that the ISBN code detects any one-digit error in an ISBN number.
4) Probably the most common error human beings make in copying long numbers is to *transpose* (interchange) two digits (for example, writing 0827404657 instead of 0824704657–do you see where the transposition occurred?). Show that the ISBN code also detects any transposition error of one pair of digits. Is it always possible to *correct* a transposition error in an ISBN number? Why or why not?

E) In addition to the situation we discussed in class where an error in a received word changes a 0 to a 1 or a 1 to a 0, there are also instances in which a digit in a received word is simply not recognizable as either a 0 or a 1. In this case, coding theory people say that an *erasure* has occurred.

1) Show that if a code has minimum distance d and a combination of t bit errors and e erasures occurs in a received word, then there is a unique closest codeword to the received word if $2t + e < d$.
2) Which are *easier* to correct, errors or erasures? Why?

Day 2 – Linear Codes. Most of the codes we will work with this summer are linear codes–codes where the set C of codewords is a vector subspace of $\mathbb{F}_2^n$. We will always the following notation:

$$\begin{cases} n & \text{is the block length} \\ k & \text{is the dimension, or information bits per codeword} \\ d & \text{is the minimum distance} \\ G & \text{is a generator matrix (rows form a basis for } C \\ H & \text{is a parity check matrix with } GH = 0 \end{cases}$$

A) For each of the following sets, give the parameters $[n, k, d]$ of the linear code $C = \langle S \rangle$, give a generator matrix, a parity check matrix, and find a basis for the dual linear code $C^{\perp}$.

1) $S = \{110100, 101101, 110111\}$
2) $S = \{1111111, 0101010, 1010101, 1110011\}$

B) Prove Theorem 2.3.19 from the text. (Hint: think about building up the basis one vector at a time.)

C) Exercise 2.6.12 from the text.

D) Prove Theorem 2.7.6 from the text.

Day 3 – Syndrome Decoding. A big advantage of linear codes (as opposed to general codes) is that they admit a simpler decoding method, called *syndrome decoding*. This is based on the fact that for a linear code with parity check matrix H, if c is a codeword and e is an error vector, then $(c+e)H = cH+eH = 0+eH = eH$. This has the following important interpretation: the information needed to correct the error e is contained in the *syndrome* $s = eH$, which depends only on the error, and not on the actual codeword. The syndromes are in one-to-one correspondence with the cosets $e + C$, and for errors e of weight at most $t = \lfloor \frac{d-1}{2} \rfloor$, the likeliest error is the minimum weight vector e in $e + C$ (the "coset leader").

A) Prove Theorem 2.10.3 from the text.

B) Prove Theorem 2.11.4 from the text.

C) Exercise 2.10.7, part c from the text.

D) Exercise 2.10.8, parts a,b from the text.

E) Exercise 2.11.21 from the text.

Day 4 – Bounds on Parameters of Codes, Hamming Codes. "Good" codes are ones for which the number of codewords (hence k) is big, but the minimum distance d is also large, so many errors can be detected or corrected. Unfortunately, it is not easy to achieve these goals simultaneously–these properties are mutually contradictory to some extent. So the search for good codes is a "balancing act," so to speak. We have introduced several bounds on the parameters of codes that give some limitations on what can be achieved:

- *Hamming (sphere packing) bound*: If C is a code of length n and minimum distance $d = 2t + 1$ or $d = 2t + 2$, then
$$|C| \leq \frac{2^n}{\sum_{i=0}^{t} \binom{n}{i}}$$
(this is valid for nonlinear codes too).
- *Singleton bound*: If C is a $[n, k, d]$ linear code, then $d \leq n - k + 1$.
- *Gilbert-Varshamov bound*: There exists a linear $[n, k, d]$ code if
$$2^{n-k} > \sum_{i=0}^{d-2} \binom{n-1}{i}.$$

These (as well as a number of other bounds) essentially give restrictions on how good codes can be. One important thing to realize is that it is not the case that codes achieving the bound (i.e. making the inequality an equality) always exist. Sometimes the bounds are not "tight."

A) Exercises 3.1.5 and 3.1.18 from the text.
B) Exercise 3.1.19 from the text.
C) Exercise 3.3.5 from the text.
D) Exercise 3.3.7 from the text (see §2.8.)
E) The form of the Gilbert-Varshamov inequality given Theorem 3.1.13 was proved by adding rows one by one to produce a parity check matrix for a code of dimension k. An alternate approach is to consider adding basis vectors (or rows of the generator matrix) one by one.

1) Show that if this approach is taken, then an $[n, k, d]$ code exists as long as

$$2^{n-k+1} > \sum_{i=0}^{d-1} \binom{n}{i}.$$

2) Is the inequality from part 1 stronger or weaker than the Gilbert-Varshamov bound given above? (You may need to experiment a bit with different n, d combinations.)

Week 2 – Cyclic Codes, General Finite Fields

Day 1 – Begin Cyclic Codes. Many of the codes that are used in applications such as the CD audio system are linear codes with "additional structure" that allows for efficient encoding and decoding. One important example is the class of *cyclic codes.* We will focus mainly on these for the remainder of the seminar. The most basic way to define a cyclic code is this: a linear code is cyclic if the set of codewords is *closed under cyclic permutations of the codeword entries.* We have seen that the algebra of polynomials is very useful to describe these codes. This is true because if we make the digits of a codeword c of length n the coefficients of a polynomial $c(x)$ of degree $\leq n-1$, then a cyclic permutation is the same as computing $x \cdot c(x)$, then taking the remainder on division by $x^n + 1$. We have also seen that cyclic codes are determined by their *generator polynomials* – the generator polynomial is the polynomial corresponding to the nonzero codeword of smallest degree. Today, we want to begin by practicing on some polynomial arithmetic with coefficients mod 2, then apply this to study some cyclic codes.

A) Exercises 4.1.19 b,c, 4.1.20 b, 4.1.21 b,c from the text.

B) Exercises 4.1.22 from the text.

C) Exercise 4.2.20 c,g (Use the polynomial forms of the codewords and the Euclidean Algorithm for the gcd. See Appendix A in the text and the class notes).

D) Exercise 4.2.22 from the text. (Note $\langle S \rangle$ is cyclic in each case.)

Day 2 – More on Cyclic Codes. Continuing our study of cyclic codes, we want to consider how to find generator polynomials for all cyclic codes. The most important fact here is the generator for a cyclic code of length n must be a divisor of $x^n + 1$. Hence we will study the irreducible factorization of $x^n + 1$ in $\mathbb{F}_2[x]$. We will always assume $\mathbb{F}_2 = \{0, 1\}$ and n is odd in the following description of binary cyclic codes.

Cyclic codes can also be described by their *idempotent polynomials.* An idempotent is a polynomial $p(x)$ satisfying $(p(x))^2 \equiv p(x) \bmod x^n + 1$. The basic idempotents are in one-to-one correspondence with the *cyclotomic cosets* in $\{0, 1, 2, \ldots, n-1\}$. The cyclotomic coset C_i is the set

$$C_i = \{2^j i \bmod n \mid 0 \leq j \leq r-1\}$$

where $2^r \equiv 1 \bmod n$ (this is where the hypothesis that n is odd is used). For instance, with $n = 15$, we have cyclotomic cosets

$$\begin{aligned}
C_0 &= \{0\} \\
C_1 &= \{1, 2, 4, 8\} = C_2 = C_4 = C_8 \\
C_3 &= \{3, 6, 12, 9\} = C_6 = C_{12} = C_9 \\
C_5 &= \{5, 10\} = C_{10} \\
C_7 &= \{7, 14, 13, 11\} = C_{14} = C_{13} = C_{11}
\end{aligned}$$

and the corresponding basic idempotents are

$$p_0(x) = 1, p_1(x) = x + x^2 + x^4 + x^8, p_3 = x^3 + x^6 + x^{12} + x^9, \text{etc.}$$

The other idempotents are obtained as linear combinations

$$a_0p_0(x) + a_1p_1(x) + a_3p_3(x) + a_5p_5(x) + a_7p_7(x)$$

for $a_i \in \mathbb{F}_2$. A generator polynomial for the cyclic code with idempotent $p(x)$ can be obtained by computing the gcd of $p(x)$ and $x^n + 1$.

A) Verify that $x^{15} + 1 = p_0(x)p_1(x)p_3(x)p_5(x)p_7(x)$ using the polynomials above.

B) Exercises 4.3.4, 4.3.5 b,d, 4.3.6 from the text.

C) Exercise 4.4.15 a,b,c,e from the text.

D) Let $g(x)$ be the generator polynomial of a binary cyclic code C of length n, and let $1 + x^n = g(x)h(x)$.
   1) Prove that the dual code $C^\perp$ is also cyclic.
   2) What does a generator polynomial for $C^\perp$ look like? Try some examples using the data generated in problem B and see if you can find the pattern. This is also stated in the text, but try to see the pattern for yourself before you look for it.

Day 3 – Finite Fields. So far, we have considered only binary codes (that is codes over the binary alphabet $\mathbb{F}_2 = \{0, 1\}$). These are probably the most important codes in applications. However, to construct binary codes with good properties, it is helpful to use the properties of other finite fields as well. Moreover, codes over alphabets that are finite fields of size > 2 are also of interest. In particular, the BCH and Reed-Solomon codes use the properties of these larger fields. All of the finite fields we will use contain $\mathbb{F}_2 = \{0, 1\}$.

Recall the general construction from class. Given an *irreducible* polynomial $p(x)$ of degree r, the set of *remainders on division by* $p(x)$ forms a field under the usual sum of polynomials and the product given by multiplying the remainders and taking the remainder of the product.

A) Exercise 5.1.15 (give the addition and multiplication tables) and 5.1.17 a,c,d

B) Exercise 5.1.16 from the text.

C) Exercise 5.1.18 from the text.

D) Exercise 5.2.8 from the text.

E) Let N_r be the number of irreducible polynomials of degree r in $\mathbb{F}_2[x]$.

(a) Show that

$$\frac{1}{1-2t} = \sum_{n=0}^{\infty} 2^n t^n = \prod_{r=1}^{\infty} \frac{1}{(1-t^r)^{N_r}}.$$

(Hint: The left side of the second equality gives the "generating function" for the sequence $A_n = 2^n$ which counts the number of monic polynomials of degree n. The other side counts the polynomials in a different way. You will need to show what the relationship is.)

(b) (Challenge Bonus Problem) Deduce that there are monic polynomials of all degrees in $\mathbb{F}_2[x]$, hence finite fields of order 2^r, for every $r \geq 1$.

Day 4 – Hamming Codes Reconsidered; BCH Codes. We have now seen that a Hamming $[2^r - 1, 2^r - r - 1, 3]$ code can be constructed as a cyclic code with generator polynomial equal to the minimal polynomial equal to the minimal polynomial of a primitive element of a field of order 2^r. Hamming codes always have $d = 3$, so they only correct 1 bit error in each received word. One of the main reasons for following this train of thought is so we can see how to construct codes with larger d in a systematic way. The resulting codes are called BCH codes (after their discoverers – Bose, Chaudhuri, and Hocquenghem). In class we saw the construction of BCH codes with $d \geq 5$. In problem C below, you will see a way to construct such codes with even larger d.

A) Exercises 5.3.8 and 5.3.9 from the text.

B) Exercise 5.4.2 from the text.

C) In class, we saw the construction of $[2^r - 1, 2^r - 2r - 1, \geq 5]$ BCH codes. In this exercise, you will see how this construction can be generalized to yield BCH codes with *designed distance* δ as large as we like (at least provided r is large enough). The actual minimum distance will be bigger than or equal to δ, so $d \geq \delta$. The most direct way to give the definition is this. Given $\delta = 2t + 1$, choose r so that $2^{r-1} > t$, and let β be a primitive element of the field $\mathbb{F}_{2^r}$ constructed using some irreducible polynomial of degree r. Let $m_s(x)$ be the minimal polynomial of β^s. Then we define $BCH(2^r, \delta)$ to be the binary cyclic code of length $n = 2^r - 1$ with generator polynomial

$$g(x) = \operatorname{lcm}(m_1(x), m_3(x), \ldots, m_{2t-1}(x)).$$

In some cases, a power β^j might be a root of one of the previous factors. If that happens, then the lcm will already contain the minimal polynomial of that β^j, so the number of factors could be smaller than indicated above.

1) Show that the dimension of $BCH(2^r, \delta)$ is $k = 2^r - 1 - \deg(g(x)) \geq n - rt$.
2) Show that the roots of $g(x)$ contain $1, \beta, \beta^2, \ldots, \beta^{2t}$.
3) Deduce that the minimum distance of $BCH(2^r, \delta)$ is at least δ. (Hint: Consider the rows of a parity check matrix and use Vandermonde determinants.)

D) Generalize the previous problem to show the following result called the *BCH bound*: Let C be a cyclic code of length n over $\mathbb{F}_2$. Let $\mathbb{F}_{2^r}$ be the smallest extension field of $\mathbb{F}_2$ containing a primitive nth root β of 1, and assume that the set of roots of the generator polynomial of C contains a string of $\delta - 1$ consecutive powers of β:

$$\beta^{m+1}, \beta^{m+2}, \ldots, \beta^{m+\delta-1}.$$

Then C has minimum distance $\geq \delta$.

Note: The actual parameters of BCH codes are somewhat "unpredictable, but in a good way" (!) Namely, it is possible for the dimension to be strictly larger than we expect (can you see how that might happen?) Moreover, the actual minimum distance d can also be strictly larger than the designed distance δ. We will see an explicit example where this happens in the lab.

Week 3 – Reed-Solomon Codes and Decoding

Day 1 – Reed Solomon Codes. The Reed-Solomon codes are closely related to BCH codes. But unlike BCH codes, they are codes over the alphabet $\mathbb{F}_{2^r}$ for some $r > 1$. A BCH code is a *subfield subcode* of a Reed-Solomon code – the subset of the Reed-Solomon codewords containing only entries from the subfield $\mathbb{F}_2 = \{0, 1\}$. The parameters of a Reed-Solomon code are also much more predictable than those of BCH codes.

The Reed-Solomon codes $RS(2^r, \delta)$ are the cyclic codes over $\mathbb{F}_{2^r}$ with generator polynomial of the form

$$g(x) = (x + \beta^{m+1})(x + \beta^{m+2}) \cdots (x + \beta^{m+\delta-1}) \tag{1.2}$$

for some m. Almost always, we will take $m = 0$, so the zeroes of the generator polynomial are the consecutive powers $\beta, \beta^2, \ldots, \beta^{\delta-1}$ of the primitive element β. The parameters of $RS(2^r, \delta)$ are

$$[n, k, d] = [2^r - 1, 2^r - \delta, \delta].$$

Note that in particular $d = n - k + 1$ so the Singleton bound is achieved. Reed-Solomon codes are important examples of *MDS codes* (this is the name given to codes achieving the Singleton bound).

There is a second, also very illuminating, way to construct these codes as well. For the code with $m = 0$ above, let L_k denote the vector space of polynomials of degree $< k$ in $\mathbb{F}_{2^r}[x]$. Consider the *evaluation mapping*

$$\begin{aligned} ev : L_k &\longrightarrow \mathbb{F}_{2^r}^{2^r-1} \\ f &\longmapsto (f(1), f(\beta), \ldots, f(\beta^{2^r-2})). \end{aligned}$$

The image of ev is the Reed-Solomon code $RS(2^r, 2^r - k)$ with parameters $[n, k, d] = [2^r - 1, k, 2^r - k]$. Note: In the notation used in our text, we are taking $S = \mathbb{F}_{2^r} \setminus \{0\}$ here. We can also consider smaller subsets $S \subset \mathbb{F}_{2^r} \setminus \{0\}$ as the set of points where the polynomials are evaluated. The resulting codes are also called Reed-Solomon codes. The two actual RS codes used in the CD audio system are formed in this way. (The CS audio code contains quite a few ingenious technical "tricks" as well – the RS codes are not the whole story!) This way of constructing Reed-Solomon codes has been vastly generalized to produce many other interesting families of codes, including the *toric codes* that figure in several of the research project topics.

A) Exercise 6.1.6 from the text.
B) Exercise 6.2.8 from the text.

C) Exercises 6.4.16 and 6.4.17 from the text.

D) Show directly that the image of the evaluation mapping *ev* coincides with the Reed-Solomon code. Hint: Consider the codewords corresponding to the basis $\{1, x, x^2, \ldots, x^{k-1}\}$ for L_k and show that, in polynomial form, each is divisible by $g(x)$ from (1.2). But be careful, this is tricky because polynomials are being used in these constructions in two different ways, and you will need to keep them straight in your thinking!

Days 2, 3–The Euclidean Algorithm (Sugiyama) Decoder. One of the reasons that Reed-Solomon codes have been used frequently for real-world applications is that several *very efficient* algebraic decoding algorithms exist for them. (These methods are *much more efficient* than the syndrome decoding we discussed in the first week would be, for instance. We will consider the method called the Euclidean algorithm (Sugiyama) decoder. (Our text discusses a different method called the Berlekamp-Massey algorithm which is comparable in efficiency but somewhat more complicated to describe.) However, it is interesting that both of these methods start from (essentially) the same *key equation.* Let $\delta = 2t + 1$ and consider the $RS(2^r, \delta)$ code (taking $m = 0$ in the text's formulas). Given a received word $w(x)$ with a error of weight t or less, we begin by computing the *error syndromes*

$$s_1 = w(\beta), s_2 = w(\beta^2), \ldots, s_{2t} = w(\beta^{2t})$$

and forming the *syndrome polynomial*

$$\Sigma(x) = s_1 + s_2 x + \cdots + s_{2t} x^{2t-1}.$$

(Recall that the generator polynomial for $RS(2^r, \delta)$ has $\beta, \beta^2, \ldots, \beta^{\delta-1} = \beta^{2t}$ as roots. So $s_i = 0$ for all i if and only if $w(x)$ is a codeword.)

The Euclidean algorithm decoder produces polynomials $\Lambda(x), \Gamma(x), \Omega(x)$ satisfying the so-called *key equation for decoding*

$$\Lambda(x)\Sigma(x) = \Omega(x) + x^{2t}\Gamma(x). \tag{1.3}$$

The polynomial

$$\Lambda(x) = \prod_{i=1}^{\tau}(1 - \beta^{e_i} x)$$

has degree $\tau \le t$ and is called the *error-locator*, since its roots $x = \beta^{-e_i}$ determine the error locations. Ω is the *error-evaluator* and can be used to determine the error values. $\Omega(x)$ has degree $\le \deg \Lambda(x) - 1$ and $\gcd(\Lambda(x), \Omega(x)) = 1$.

Note: The notation used in the text for the key equation is different – our $\Lambda(x)$ is called $\sigma_R(x)$ and our $\Sigma(x)$ is a *shifted version of* the book's $s(x)$, so the other polynomials in our version are slightly different. It is unfortunately pretty rare that mathematicians settle on a completely standard notation for anything. So it pays to get proficient at translating back and forth when you compare different sources(!)

A) For each of the following collections of syndromes, carry out the Sugiyama algorithm to find the error locator and error-evaluator, then determine all roots of the error locator by Chien search and the error values by the Forney Formula (if possible). If you cannot determine the error locations, say what prevented you from completing the algorithm.
   1) $s_1 = \beta$, $s_2 = \beta^2$, $s_3 = \beta^{10}$, $s_4 = \beta^6$, $s_5 = \beta^5$, $s_6 = \beta^7$, $s_7 = \beta^6$
   2) $s_1 = \beta^7$, $s_2 = \beta^{13}$, $s_3 = 0$, $s_4 = \beta^{10}$, $s_5 = \beta^{11}$, $s_6 = \beta^2$, $s_7 = \beta^4$

B) Show that if $\Lambda'(x), \Omega'(x), \Gamma'(x)$ are another triple of polynomials satisfying (1.3) for the same $\Sigma(x)$ and $\deg \Lambda'(x) \leq t$, $\deg \Omega'(x) \leq \deg \Lambda'(x) - 1$, and $\gcd(\Lambda'(x), \Omega'(x)) = 1$, then there is a nonzero constant $\gamma \in \mathbb{F}_{2^r}$ such that $\Lambda'(x) = \gamma\Lambda(x)$, $\Omega'(x) = \gamma\Omega(x)$, and $\Gamma'(x) = \gamma\Gamma(x)$.