Multiple Error Detection and Correction by CRC

Prof. Dr. W. Kowalk
University of Oldenburg
kowalk@kowalk.de

Abstract

It is well known that single bit error detection and correction can be done by CRC (*cyclic redundancy check*). A canonical extension of this method to detect and correct any number of errors is theoretical known, however, only rarely used. Intention of this paper is to remind the readers to the theoretical and practical foundations of these methods.

Introduction

Error detection is important for many network applications. Although nowadays transmission over fibre optics is highly reliable, there are now many more wireless applications, which are much more error-prone than wired transmissions; also other less reliable carriers like high voltage systems (dLAN) are in common use nowadays. In many applications ARQ is used to repeat a block of data that has been discovered to be incorrect. This is not very efficient, and in some applications even infeasible, e.g. when transmission time is very long, or when erroneous data are read from a read only memory (ROM). Therefore forward error correction is useful for many applications; provided it is not too expensive it should be used on a regular basis.

We show in this paper, how error correction can be done with Cyclic Redundancy Check (CRC), extending this approach to detection and correction of multiple bit errors in a systematic way, as follows from theoretical considerations (*bhargava*). This paper presents at first the well known and ubiquitously applied basic approach, together with some formal notations, and extend this to multiple error correction. The effort for this is proportional to the effort for a single bit error correction times the number of bits to be corrected. Some further advantages are also derived and discussed.

The Algebra

We assume the reader is familiar with the algebraic concepts of CRC and repeat here our notation and the main properties. A more systematic presentation of the algebra is given in appendix 6.

CRC is a cyclic block code, usually of optimal length. It is well known as BCH (*bhargava*). Its main properties are that errors can be detected and corrected, where the latter property depends on the overhead.

A sequence of bits is represented as a polynomial over a field (0,1); e.g. the sequence G=10011 can be represented as $G(X)=X^4+0\cdot X^3+0\cdot X^2+X^1+X^0=X^4+X+1$, or H=101000011 as $H(X)=X^8+X^6+X^1+X^0$; obviously bit sequences are low endings. Polynomials can be added, e.g.

$$G(X)+H(X)=X^4+X+1+X^8+X^6+X+1=X^8+X^6+X^4,$$

since 1+1=0 in the field (0,1). The degree of a polynomial is the greatest non-zero exponent, i.e. the degree of $G(X)$ is 4 and the degree of $H(X)$ is 8. A polynomial with no terms is written as $R(X)=0$ and has no degree; $R(X)=X^0=1$ has degree 0; $R(X)=X+1$ has degree 1, etc.

Multiplication of polynomials is written by ·, where · is distributive over +; thus we have $(X^4+X+1)\cdot(X+1)=X^5+X^2+X+X^4+X+1=X^5+X^4+X^2+1$. To 'shift' all terms by k bits, we can multiply by X^k, for example with $k = 3$, we get $G(X)\cdot X^3=X^{4+3}+X^{1+3}+1\cdot X^3=X^7+X^4+X^3$ or 10011000.

We use $\perp$ as operator for the remainder of division, so $R(X)=D(X)\perp G(X)$ means that dividing $D(X)$ over $G(X)$ yields the remainder $R(X)$. We only are interested in remainders, never in quotients $Q(X)$. The exact condition for $R(X)$ in the last formula is $D(X)=Q(X)\cdot G(X)+R(X)$, where the degree of $R(X)$ is less than that of $G(X)$; $Q(X)$ and $R(X)$ are uniquely determined by $G(X)$ and $D(X)$.

We call a single power expression X^k for arbitrary *k* a *1-term* and sometimes count the number of 1-terms in a polynomial. We call a polynomial *even*, if it has an even number of 1-terms, *odd*, otherwise. Also $R(X)=0$ is an even polynomial with no degree and zero terms.

Single Bit Error Detection and Correction

We repeat here the well known method to detect and correct a single bit error. These techniques are widely used, e.g. in protocols like Bluetooth or ATM. We start with detection of a single bit error.

Let $G(X)=X^n+\ldots+X^m+\ldots+1$ be a polynomial of degree *n*, called *generator*. To detect and correct a single bit error $E(X)=X^k$ in a data stream $D(X)$, we send the data stream with appended remainder $R(X)=(D(X)\cdot X^n)\perp G(X)$ to the receiver: $N(X)=(D(X)\cdot X^n)+R(X)$; from distributivity of $\perp$ over + follows

$$N(X)\perp G(X)=(D(X)\cdot X^n)\perp G(X)+R(X)\perp G(X)=R(X)+R(X)=0.$$

In case of a single bit error $E(X)=X^k$ the receiver gets the data $N(X)+E(X)$; the receiver performs the same division

$$(N(X)+E(X))\perp G(X)=(N(X)\perp G(X))+(E(X)\perp G(X))=0+E(X)\perp G(X)=S(X).$$

In case of a single bit error the remainder $S(X)$ is never zero. From the last formula we see that $S(X)$ is uniquely determined by $E(X)$ and $G(X)$, and neither by $D(X)$, $N(X)$ or anything else. $S(X)$ is also called *syndrome*.

Single bit error correction

Now we want to correct a single bit error, where the only information available to the receiver is $S(X)$ and the mutual generator $G(X)$. Thus the number of different bit errors detectable by this method is bounded by the maximum number of different remainders.

If the degree of $G(X)$ is *n*, the degree of the remainder $S(X)$ is less than *n*, so that there are at most *n* bits in $S(X)$. Thus there are at most 2^n distinct $S(X)$, one of which is 0. Thus we can expect not more than 2^n-1 single bit errors to be distinguishable by this method. Such a generator, which can distinguish 2^n-1 distinct 1-bit errors is called an *optimal generators of degree n and order 1*.

If there is an odd number of 1-terms in $G(X)$ (or $G(X)\perp(1+X)=1$), then the remainder $S(X)$ can be 0 or have any other combination of bits. Thus there may be (and for all degrees are) optimal generators so that there are distinct remainders for all 2^n-1 single bit errors $E(X)=X^k_{k<2^n-1}$, i.e. $1,X,X^2,\ldots X^{2^n-2}$. However, these generators cannot detect all odd numbers of bit errors, for example the pattern of $G(X)$, which is the main reason why this type of generator is usually not used.

The remainders of $G(X)$ can be distinguished by $2^{n-1}-1$ (plus 0) remainders with an even number of terms and 2^{n-1} with an odd number of terms, one of which, however, can never be the result of division of any X^k by $G(X)$; it can easily be constructed by $G(X)$, and will be exposed in appendix 4.

If we assume that there is an even number of 1-terms in $G(X)$ (or $G(X)\perp(1+X)=0$, i.e. $G(X)$ has the factor 1+X), then follows from simple considerations that in case of a single bit error there

is an odd number of bits in the remainder $S(X)$, i.e. $S(X)$ is odd; there are at most $2^{n-1}-1$ such remainders, i.e. we cannot correct more than this number of different single bit errors. Thus we have to select a generator $G(X)$ that produces distinct remainders for all $E(X)=X^k_{k<2^{n-1}-1}$. There are several of such $G(X)$ for most degrees n, (for all degrees that we have analyzed, i.e. 5 to 32) some of which we list in appendix 1. These generators are called *optimal even generators of degree n and order 1*.

It should be clear that this number $2^{n-1}-1$ encloses the n bits for the remainder, so that the effective payload that can be corrected is $2^{n-1}-n-1$ bits.

The *period* of G is defined as the smallest p, so that $X^p \perp G(X)=1$. Thus an optimal even generator of degree n has the period $p=2^{n-1}-1$. Generators of this type can be verified by simple tests of all remainders of single bit errors $1 \ldots X^{2^{n-1}-2}$. In appendix 5 we verify formally that all such remainders of a generator with period $p=2^{n-1}-1$ are different.

Besides all single bit errors, this type of generator can detect all odd number of bit errors (but not correct them for three or more bit errors). An even generator $G(X)$ of degree n is optimal, if and only if the division procedure of $X^{2^{n-1}-1}$ by $G(X)$ yields a one bit intermediate expression X^k only for k=0, and thus leaves remainder 1. This follows from bit filter theory (*kowalk*). A simple proof for this statement can be found in appendix 5.

Thus a test for any generator to be optimal can be performed by not more than 2^{n-1} xor-additions of terms, provided the generators are coded as integer numbers. The bit error $X^{2^{n-1}-1}$ leaves the remainder $X^0=1$, so the two bit errors $X^{2^{n-1}-1}$ and 1 cannot be distinguished; thus only those $2^{n-1}-1$ bit errors $X^0=1, X, \ldots, X^{2^{n-1}-2}$ can be corrected by inspecting the remainder $S(X)$.

An optimal generator is a *prime* (also *primitive* or *irreducible*) polynomial (or a prime polynomial times $1+X$ for even generators), but in general not all prime polynomials are optimal generators.

Since $X^{2^{n-1}-1}$ leaves the remainder 1, $1+X^{2^{n-1}-1}$ can be divided by $G(X)$. However, if $G(X)$ is not optimal, there is another minimal exponent $p>0$, so that $1+X^p$ can be divided by $G(X)$. Thus because of $q=2^{n-1}-1-p$, $G(X)$ divides $X^q \cdot (1+X^p)=X^q+X^{2^{n-1}-1}$ as well. Thus, X^q and $X^{2^{n-1}-1}$ leave the same remainder, which is of course not 0, so that the position of the erroneous bit cannot be determined uniquely. If the data block to be transferred is always shorter than the minimal period q, also non optimal generators can be used, perhaps with other advantages. It shall be mentioned here as well without detailed proof that with a generator of period p all two bits errors X^r+X^s in a block of length less than p can be detected safely, thus never leave a remainder 0.

Two Bit Error Correction

To correct two bit errors $E(X)=X^j+X^k$ a similar method as for one bit errors is applicable. We present here a CRC-method which uses two distinct generators $G_1(X)$ and $G_2(X)$ and compute two remainders $S_1(X)$ and $S_2(X)$. If there are two generators $G_1(X)$ and $G_2(X)$ such that for any combination of two bit errors $(X^j, X^k)_{j<k}$ there is a unique combination of remainders $(S_1(X), S_2(X))$, then it is possible to determine the indices (j,k) from $(S_1(X), S_2(X))$. This can be tested by corresponding simulations to find such *pairs of optimal generators of order 2*. Also some of these pairs of generators can be found in the appendix 2, together with an explicit demonstration of the method for two even generators of degree $n=5$.

The number of possible error combinations is $p \cdot (p-1)/2 \approx p^2/2 < 2^{2n-3}$, since $p \le 2^{n-1}-1$ is the maximum length of the data block. Since the number of different remainder combinations is $(2^{n-1}-1)^2 \approx 2^{2n-2}$, it is not impossible to find such generators. Of course, this calculation has to be done only once to implement this method.

Another advantage of this method is that we can now distinguish between single and other odd numbered bit errors. Three bit errors always leave an odd number of terms in the remainder, as well as single bit errors and all other odd numbered bit errors, so that a three bit error may lead to a (one-bit)-correction, which yields completely wrong results. But now we can perform two tests for a single bit error, and if both yield the same position of the single bit error, then there is much higher chance that there was a single bit error instead of another odd numbered error. Of course, we have to use those optimal generators of order 2 that produce best chances for these properties. Usually they can distinguish uniquely between single and triple bit errors. This can be simulated as well.

Also we can test whether four or more bit errors can be detected, although not corrected (sometimes this is called *Hamming* distance). If for example all two and four bit errors and all odd errors can be detected, then at least five bit errors can be detected, although only one and two bit errors can be corrected. Again the problem might be that two and four bit errors are all leaving an even number of remainders, so that the remainders can be misinterpreted. Obviously, there is a danger to correct an error wrongly, as it is in case of single bit errors. Therefore it is useful to have a higher redundancy in the system, as the number of possible double bit errors is only half of the number of possible remainders.

More Bit Errors

It should be clear now how we can extend this method to 3 or more bit errors. For up to m bit errors we have to generate m remainders with m different generators, where simulations have to show that for all combinations of m errors there are unique combinations of remainders. We have done this for 3 bit errors, and found some of those optimal generators of order m=3 for several degrees n. Results are presented in the appendix 3.

Here again, redundancy increases, since now the number of possible triplet error combinations is only about one sixth of the possible remainder combinations. Thus many other errors can be detected, as well, although the fundamental problems are the same as for double or single errors.

Finding Error Bit Positions

If the remainders for a special error condition are found, then the error position can be looked up by a corresponding table. The method is demonstrated in appendix 2 for error correction of order 2. Here, the remainders are usually from a special class, e.g. the number of 1-terms is either even or odd (for even generators). To reduce the table size, which can become very large, it might be useful to map the possible remainders to a dense sequence of integers 0...k, and compact those tables to the minimum size required. This mapping can be done algorithmically or by another table, which is linear in the block length.

If the table cannot be implemented, since its size is bounded by $O(k^m)$, where k is the maximum block size in bits and m is the number of bit errors to recover from, then algorithms can be developed the run time of which are of a similar order $O(k^m)$. For example to find the position of a single bit error, the generator division procedure can be inverted: wipe out the lowest bit with the generator's lowest bit and proceed until there is only one bit left. This is the erroneous bit. The effort is proportional to the block length $O(k)$.

For double error bits the algorithms for one bit errors must be combined, so that either the order of memory or the run time is $O(k^2)$. There is also a combination possible, by use of a linear array of single bit errors and the corresponding remainders (for each generator). From the equation

$$X^i = Q_i(X)\cdot G_1(X) + R_{1i}(X) \text{ and } X^j = Q_j(X)\cdot G_1(X) + R_{1j}(X)$$

follows after summation $X^i + X^j = Q_{ij}(X)\cdot G_1(X) + R_{1i}(X) + R_{1j}(X)$, so that the remainders of two bit errors can be computed from the sum (or bitwise xor-operation) of the remainders of the corre-

sponding single bit errors. Thus for a single bit error X^i we can compute its remainder, and from the remainder and the common remainder of the two bit errors we can compute the other bit error X^j (all in constant time). Then we compute the two remainders with the other generator and compare their sum with the other remainder of the other double error. If it coincides, the positions of the two bit errors are found. If not, another bit error position X^i must be guessed and the procedure repeated. The effort is therefore O(k).

This method can be extended to higher number of m bit errors. Then only those bits can be valid where in all sequences the m bit positions coincide.

This method can be applied in computer communication (including telecommunications) as well as data storage. While in the first case recovery from bit errors is time critical, in the second case correctness of data is more important, so that the corresponding algorithms to find the positions of erroneous bits should be adjusted to the application area. Thus, if higher reliability is required, this method can be implemented in an efficient way for a broad area of applications.

Further Considerations

In some applications the generator can check much longer sequences than those which are used in such protocols. E.g. the length of Ethernet frames can be no longer than 12000 bits payload and 208 bits overhead. To protect those frames, generators of degree 14 with odd or degree 15 with even number of 1-terms are sufficient (maximum length including remainder is 16367 bits); however the degree of the generator of Ethernet is 32. If special applications are known, then generators may be used that are not optimal for the total length (of 16367 bits in the example), but only for the shorter length. This can easily be tested by simulation.

The generators are completely independent. Thus it is possible to apply generators of different degree at the same time so that the size of the blocks, which can be protected with CRC, will be limited to the smallest degree. For example, there are optimal generators of degree 7 and 8, which both can protect blocks of size 127 bits (if generator of degree 7 holds an odd number of 1-terms). Therefore in special situations it might be useful to use those distinct generators with different degrees.

Also, since the degree of the generators limits the size of detectable error bursts, it might be useful to use one long generator to detect long error bursts, while generators with smaller degree can be used at the same time to detect and correct two or more bit errors.

Another method is to use a single generator of doubled degree $2 \cdot n$ to correct 2 bit errors in a block of size $p=2^{n-1}-1$. The number of different error states $\left(X^i+X^j\right)_{(0\le i<j<p)}$ is less then the number of distinct remainders of optimal generator of degree $2 \cdot n$, since $\frac{p \cdot (p-1)}{2} \le \frac{\left(2^{n-1}\right)^2}{2} = 2^{2n-3} \le 2^{2n-1}-1$. Thus there might be generators that can display distinct remainders for each error state. However, simulations for $n=8$ have proved that there are no optimal generators for this method for all degrees n.

Overhead

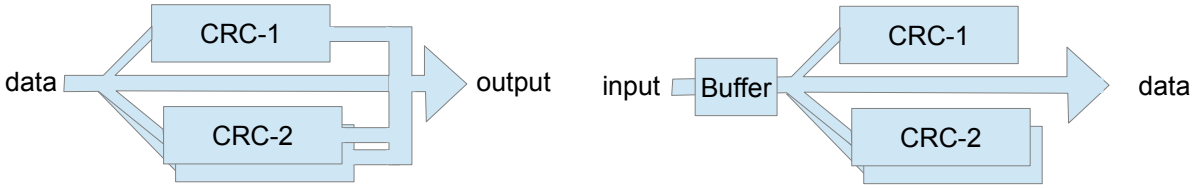
In the method presented here the overhead can become very high, if the degree of the generator is very small. For even generators with degree 5 there are not more than 10 bits payload, while the overhead is m times 5. Thus for two bit correction the overhead is 100%, for four bit correction it would be 200%; another protocol (FEC 1/3 rate) used in Bluetooth, for example, sends all bits three-times and has therefore an overhead of 200%, either. Thus in this case the FEC 1/3 rate method, as used in Bluetooth, may be more adequate, since this can correct all errors, provided that they are not clustered. But for higher degrees, e.g. degree 8, there are 119 bits payload while the

control information is 16 or 32 bits, respectively, for 2 or 4 bits correction, and thus our method is more useful in situations with low error probability when there is a high chance for clustered errors.

In case of errors in remainders only, there may be problems to discover the correct errors. If for example there is an error in only one remainder, then usually the parity of the 1-terms in the remainders are different, which can be detected and the conclusion derived, that there are errors which cannot be corrected. If, however, there are one bit errors in both remainders, then this type of error may be either undetectable or lead to wrong 'correction' results. This situation may lead to consecutive errors, which cannot be detected.

Implementation

Hardware implementation of CRC is usually sustained by a special shift register, in which the generator is hardcoded. Using two or more different shift registers, sending data is done by pushing data through all registers in parallel, and when finished appending the remainders in the registers one after the other to the data stream. This can be performed on the fly with no additional delay.



The receiver runs the data through the same registers. When the remainders arrive they are sent one after the other to the different registers. To recognize the end of the data stream and start of remainders, it might be necessary to add a small buffer with capacity of all remainders, so that the remainders can be sent to the correct registers. The content of those registers can now be used to compute the position of the erroneous bits. Additional delay is the run time through the additional buffer. Since erroneous data have to be corrected, there will be another delay for buffering of the data and repairing the erroneous bits.

Conclusions

We can specify a general technique to detect and correct more than a single bit error by Cyclic Redundancy Check, as is possible because of theoretical considerations; see for example (*bhargava*). Thus we assume for some applications this type of error correction can be useful. It depends on the error structure and the required number of errors to correct. However, other methods, like classical Reed-Solomon, should be considered as well, since they may perform better in some situations.

Appendix 1

Optimal generators of order 1

Optimal generators of degree 5 are X^5+X^3+X+1, and $X^5+X^4+X^2+1$.

Optimal generators of degree 6 are X^6+X^2+X+1, $X^6+X^3+X^2+1$, $X^6+X^4+X^3+1$, $X^6+X^5+X^4+1$, $X^6+X^5+X^3+X^2+X+1$, and $X^6+X^5+X^4+X^3+X+1$.

Optimal generators of degree 7 are $X^7+X^4+X^2+1$, X^7+X^5+X+1, $X^7+X^5+X^3+1$, $X^7+X^6+X^2+1$, $X^7+X^5+X^4+X^2+X+1$, and $X^7+X^6+X^5+X^3+X^2+1$.

Some optimal generators of degree 8 are X^8+X^2+X+1, X^8+X^4+X+1, $X^8+X^4+X^3+1$, $X^8+X^5+X^3+X^2+X+1$, and $X^8+X^7+X^6+X^5+X^4+X^3+X^2+1$.

There are 1800 optimal even generators of degree 16. Some of them are $X^{16}+X^2+X+1$, $X^{16}+X^5+X^4+X^3+X+1$, $X^{16}+X^8+X^6+X^5+X^4+X^3+X+1$, $X^{16}+X^{11}+X^{10}+X^7+X^6+X^5+X^4+X^3+X+1$, $X^{16}+X^{15}+X^{14}+X^{13}+X^{12}+X^{11}+X^{10}+X^4+X^2+1$, and $X^{16}+X^{15}+X^{14}+X^{13}+X^{12}+X^{11}+X^{10}+X^9+X^8+X^7+X^6+X^5+X^4+X^2+X+1$.

Remainders of X^k with optimal generator X^5+X^3+X+1 are

k =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Remainder =	00001	00010	00100	01000	10000	01011	10110	00111	01110	11100	10011	01101	11010	11111	10101

Remainders of X^k with optimal generator X^6+X^2+X+1 are

k =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Rem =	1	10	100	1000	10000	100000	111	1110	11100	111000	110111	101001	10101	101010	10011	
k =	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Rem =	100110	1011	10110	101100	11111	111110	111011	110001	100101	1101	11010	110100	101111	11001	110010	100011

Appendix 2

Examples for Generators of Order 2

The following table demonstrates the method developed here for generators of degree 5 and for 2 bit errors, i.e. order 2. The generators $G_1=X^5+X^3+X+1$ and $G_2=X^5+X^4+X^2+1$ yield some remainders for any bit combination X^i+X^j, and you can find the two erroneous bits in the corresponding rows and columns. Only remainders with 2 or 4 1-terms can be generated.

Remainder of Generator 101011							Remainder of Generator 110101								
	00011	00101	00110	01001	01010	01100	01111	10001	10010	10100	10111	11000	11011	11101	11110
00011	1,0	14,6					7,2			13,9	10,4	11,8		5,3	
00101	8,5	2,0	11,3	7,1	14,4			13,12				10,6			
00110		12,9	2,1		7,0		6,4						11,5	14,1	8,3
01001	14,9			3,0	11,2				13,6		5,1		12,1	8,7	
01010		11,7	9,6		3,1	12,4			8,2	5,0	14,13				
01100	12,6			9,4	13,1	3,2	11,0			8,1					7,5
01111	7,3		14,12			11,1		5,2			8,0			13,4	10,9
10001		10,1		12,5				4,0		7,6	11,9		13,8		14,2
10010	13,11		10,0				9,8	14,7	4,1	12,3			6,2		
10100					12,8	10,7		9,3		4,2		13,5	14,0	6,1	
10111	10,2	13,3				9,5			12,11			14,1	7,4		6,0
11000			13,7	8,6					10,5	14,11		4,3		9,2	12,1
11011		5,4		13,2		14,8	10,3	11,6	9,7					12,0	
11101					6,5		13,1	10,8	14,3		12,7	9,0			11,4
11110			8,4	11,1		13,0	14,5				6,3	12,2	9,1		

For example the two bit errors $E(X)=X^8+X^6$ leave the remainders 11000 and 01001, where we find the entry 8,6, which means that there are two error bits at position 6 and 8. Another example is the error $E(X)=X^{14}+X^{13}$ with remainders 01010 and 10111, thus the errors are in bits 13 and 14. There are 105(=15·14/2) different combinations, which can be checked all manually. The table size is 225(=15^2), so that there are 120 empty fields. This shows that the table is not optimally compressed.

The empty fields represent no valid combinations of remainders of two bit errors. Thus there is some redundancy in the method, so that other error types (like some 4-bit errors) might be detectable.

Other combinations of optimal generators of order 2 are

(X^8+X^2+X+1 with X^8+X^4+X+1) or

($X^8+X^6+X^5+X^4+X^3+X^2+X+1$ with $X^8+X^7+X^6+X^5+X^4+X^3+X^2+1$), and

($X^{16}+X^2+X+1$ with $X^{16}+X^{11}+X^2+1$).

Appendix 3

Some triplets of optimal generators of order 3 are

$(X^8+X^4+X^3+1,\ X^8+X^4+X+1,\ X^8+X^2+X+1)$ and

$(X^{10}+X^8+X^5+1,\ X^{10}+X^5+X^2+1,\ X^{10}+X^3+X^2+1)$.

Appendix 4

If there is a Generator $G(X)=\sum_0^n G_i \cdot X^i$ of degree n with even number of 1-terms, we can construct a remainder $H(X)=\sum_0^{n-1} H_i \cdot X^i$ of degree less than n, so that $(H(X)\cdot X)\perp G(X)=H(X)$. We do this by solving the linear equation $H_i=H_{i+1}+G_{i+1}$, by setting $H_n=0$, since H's degree must be lower than n. For example, if $G=101000011$, then follows $H=11000001$. From this follows the condition from above. Obviously this is the uniquely determined remainder, the pattern of which is never changed by $G(X)$, so it can never become a 1-term. This is not possible with an odd number of 1-terms in $G(X)$, since such generators change the parity of the number of the terms in each step during division.

We can invert the procedure, by 'undividing' with $G(X)$, that is starting with the remainder and add $G(X)$ in that way, that the lowest bit of the remainder disappears, etc. If the resulting pattern changes, we may eventually come to a 1-term X^k. This procedure is of course unique and reversible, so that in the case above no 1-term can have the remainder $H(X)$. This is possible for all generators with even number of 1-terms at least once, although similar phenomenons can occur several times in other non-optimal generators.

Appendix 5

If there are two different bit errors X^r and X^s ($r<s$) with the same remainder $R(X)$, then $X^r+X^s=(X^{s-r}+1)\cdot X^r$ can be divided by $G(X)$, since

$X^r=Q_r(X)\cdot G(X)+R(X)$ and $X^s=Q_s(X)\cdot G(X)+R(X)$ yields after addition

$X^r+X^s=(Q_r(X)+Q_s(X))\cdot G(X)+R(X)+R(X)=(Q_r(X)+Q_s(X))\cdot G(X)$.

Since $G(X)$ cannot divide X^r (since $G(X)$ has the term 1), it must divide $X^{s-r}+1$, so that $s-r\geq p$ is greater than or equal to the period p of the generator $G(X)$, as defined above. Thus if $s<p$ then there is no positive r, so that X^r leaves the same remainder as X^s.

If a remainder is given, we can compute the single bit error X^r with this remainder, by 'undividing' the remainder. For details see appendix 4.

Appendix 6

The field (0,1) has two elements {0,1} and two commutative operators + and ·, where 0+0=1+1=0, 0+1=1+0=1; and 1·1=1, 0·0=0·1=1·0=0. Since 1+1=0 follows 1=0–1=–1. Thus addition and subtraction is the same in this field. Division is only possible with 1 as divisor, thus 0/1=0, 1/1=1.

The polynomials over this field have the common form $\sum a_i \cdot X^i$, where terms with $a_i=0$ are not displayed and terms with $a_i=1$ are displayed without the factor, e.g. $P(X)=X^5+\ldots+X^2+X+1$; the *degree* of a polynomial is the largest exponent of all terms, e.g. 5 in the last example. From the properties of the field (0,1) follows $X^r+X^r=(1+1)\cdot X^r=0$. In the set of all polynomials summation and multiplication are specified. Addition is associative, thus polynomials are summed term by

term. Multiplication of two terms is defined as $X^r \cdot X^s = X^{r+s}$; multiplication is distributive over addition, so that each term of a sum is multiplied by each factor of the other sum, and then all terms are summed, e.g. $(X^4+X+1)\cdot(X+1)=(X^5+X^2+X)+(X^4+X+1)=X^5+X^4+X^4+1$, since $X+X=0$.

Division of P by G is defined as $P(X)=Q(X)\cdot G(X)+R(X)$ and written as $P(X)/G(X)=Q(X)$ remaining $R(X)$, where the degree of R is less than that of G. To compute Q and R, we use the following algorithm: Let $Q(X)=0$, then

while (k=([degree of P]–[degree of G])≥0) $\{Q(X)+=X^k;\ P(X)+=G(X)\cdot X^k\}$.

Then follows $P(X)=Q(X)\cdot G(X)+R(X)$ and the degree of R is less than that of G.

If P is an arbitrary polynomial, then $P(X)=Q(X)\cdot(X+1)$ has an even number of terms; we call this polynomial also *even*. An even polynomial P has the factor $X+1$, i.e. $P(X)=Q(X)\cdot(X+1)$ with remainder $R(X)=0$. This follows from the division procedure, since the division algorithm either removes the two highest terms X^k+X^{k-1}, if they are both one, or it shifts the highest term one position to the right, if the next highest term is 0. Thus all adjoining pairs of two terms are removed, and only those; a single term leaves always the remainder 1. From this follows that a polynomial has an even number of terms, if and only if it has the factor $X+1$.

A polynomial can be *prime* (also *primitive* or *irreducible*), which means it has no other factor than 1 and itself. Examples for prime polynomials are X^3+X+1 or $X^9+X^7+X^2+X+1$. The factors of a non-prime polynomial are uniquely defined, e.g. $X^4+X^3+X+1=(X+1)^2\cdot(X^2+X+1)$. Obviously, an even polynomial can never be prime, since it has the factor $X+1$. We call an even polynomial *even-prime*, if it has only one factor $1+X$ and at most one further factor; e.g. the last example is not prime, since the polynomial X^4+X^3+X+1 has twice the factor $X+1$, however $X^8+X^6+X+1=(X+1)\cdot(X^7+X^6+1)$ is even-prime; we also use the shorter form *prime* instead of even-prime in case of an even polynomial.

In this paper we discuss *optimal* generators, which are all prime (or even-prime), however, not all prime polynomials are optimal. Thus the property of being prime is necessary, but not sufficient for a polynomial to be optimal.

Literature

Artin: „Galoissche Theorie“ Deutsche Taschenbücher (1959).

Bhargava: „Forward Error Correction Schemes for Digital Communication“ IEEE Communication Magazine 1983.

Koblitz: „Algebraic Aspects of Cryptography“. Algorithms and Computation in Mathematics, Vol. 3. Springer (1997).

Koopmann: „32-Bit Cyclic Redundancy Codes for Internet Applications“ Preprint of a regular paper to appear in The International Conference on Dependable Systems and Networks (DSN) 2002.

Kowalk: Bitfilter. Link: „*einstein.informatik.uni-oldenburg.de/papers/**Bitfilter**.pdf*“¶