Preserving the integrity of cyclic-redundancy checks when protected text is intentionally altered

by D. R. Irvin

As a digitally encoded message traverses a series of point-to-point communication links, it may be necessary to change the contents of that message at an intermediate station. If bit errors are introduced by the intermediary while the text is unprotected, these errors will be subsequently undetectable by cyclic redundancy checks. An algorithm is presented here for ensuring that such errors will not go undetected. Since the cyclic redundancy check is based on a linear mathematical operation, the frame-check sequence may be modified, rather than recalculated, by each intermediary changing the protected text. A frame-check sequence constructed in this way will reveal any errors introduced in the transmission path when the message is finally examined at the ultimate destination. Examples of the proposed technique applied to various local-area network bridges are developed. The technique is shown to be beneficial in these examples when the internal bit-error ratio of the text-changing device exceeds 10^{-19} on unprotected paths.

1. Introduction

Cyclic redundancy check (CRC) algorithms are often used in digital communication systems to detect the presence of errors introduced during transmission. Frame-check sequence (FCS) bits are generated by the CRC at the transmitter and appended to the outgoing text. The receiver performs an equivalent CRC on the incoming text and compares the resulting FCS to that

appended by the transmitter. If the FCSs match, then the likelihood of error-free transmission is ensured with a high probability.

The protected text may be forwarded from its source to its destination along a number of point-to-point links. In simply structured systems, the intermediaries forward the text unchanged. In certain more complex systems of current interest, such as bridged local-area networks and frame-relay systems, it may be necessary for the intermediate stations to change the text of a message as it is passed from point to point. For example, a frame-relay station may remove its own address from a protected transmission header and insert the address of the next node before passing the message on. When such a change is made, the original FCS, of course, is no longer valid. The standard procedure is to recalculate the FCS based on the new data. If an error occurs within the intermediate station, however, the new FCS will be based on corrupted text, and from then on the error will be undetectable. Thus, the reception of a seemingly valid FCS at the destination no longer serves to ensure correct end-to-end transmission.

An algorithm has been devised to ensure end-to-end integrity in transmission systems that require protected text to be altered at intermediate stations. The algorithm specifies a means of modifying the original FCS, as opposed to recalculating it, each time the protected text is changed. Modifying the FCS in step with changes in the protected text maintains end-to-end integrity to the extent supported by the inherent limits of the particular CRC in use.

The CRC algorithm is based on a linear mathematical operation. As such, it is readily manipulated by application of the principle of superposition. The linearity of the basic CRC algorithm should be well known to mathematicians familiar with the algebra of polynomials having coefficients from a finite field, but current literature indicates that designers of communication systems and protocols may be unacquainted with these ideas [1]. Perhaps this may be attributed to the fact that industry-standard CRCs include steps beyond basic polynomial division; these operations corrupt the linearity of the techniques.

In the body of this paper, the algorithm for modifying the CRC is given, and the assertions made here are justified. An example of the technique for preserving end-to-end CRC integrity applied to a bridge interconnecting an ANSI-standard FDDI local-area network (LAN) and an IEEE Standard 802.5 token-ring LAN is developed in detail. Interconnecting IEEE Standard 802.4 (token-bus) and 802.5 or FDDI LANs is discussed briefly. The technique is shown to be beneficial in bridges and other text-changing devices with internal bit-error ratios exceeding 10^{-19}.

2. Cyclic redundancy checks

• *Preserving end-to-end protection*

The frame-check sequence (FCS) may be thought of as a second version of the transmitted text $I(x)$, although, of course, the text may not be reconstructed from the FCS. The existence of two versions of the text is the key to the problem of providing end-to-end integrity. Suppose that an intermediate station makes an intended change in the text, creating $I'(x)$, and then introduces an unintended error while the text is unprotected, inadvertently creating $I''(x)$. If this station now discards $R(x)$, the original FCS, and then calculates $R'(x)$, a new FCS, both versions of the original text, namely $I(x)$ and $R(x)$, are lost. From then on, the newly introduced error will be undetectable. If, instead of recalculating the FCS, the intermediate station modifies the existing FCS to reflect the intentional bit changes, the two versions of the new text, $I''(x)$ and $R'(x)$, no longer agree. The error introduced by the intermediate station is easily detected.

• *Mathematical description*

The reader is assumed to be familiar with the technique of representing an N-tuple of bits by a polynomial of degree $N-1$ having coefficients drawn from a finite field (the field consisting of 1 and 0, in this case), and with the laws of addition, multiplication, and division for these polynomials [2]. Let $I(x)$ be the polynomial representing the information to be transmitted, and let $G(x)$—of degree n—be the generating polynomial for CRC. In practice, $I(x)$ is multiplied by x^n [i.e., shifted, to make room for appending the n-bit frame-check sequence (FCS)]. The frame-check sequence, $R(x)$, and the information polynomial, $(x^n)I(x)$, are congruent with respect to the generating polynomial, $G(x)$.

The division showing congruence may be written as follows:

$$(x^n)I(x) = Q(x)G(x) + R(x). \tag{1}$$

$Q(x)$ is the quotient, and is of no further interest here. Adding $Q(x)G(x)$ to both sides of Equation (1) gives (for the case of the finite-field coefficients considered here)

$$R(x) = (x^n)I(x) + Q(x)G(x).$$

This is equivalent to the statement of congruence:

$$R(x) = (x^n)I(x) \qquad [\mathrm{mod}\, G(x)].$$

• *Altering the protected text*

Suppose that bits within $(x^n)I(x)$ corresponding to the nonzero terms of some polynomial $P(x)$ are to be intentionally changed by an intermediate station. The new text, $(x^n)I'(x)$, is generated by adding $(x^n)I(x)$ and $P(x)$ under the rules of modulo-2 addition:

$$(x^n)I'(x) = (x^n)I(x) + P(x).$$

Let $R''(x)$ be the frame-check sequence for $P(x)$. Then, the new FCS to be transmitted is also found by addition:

$$R'(x) = R(x) + R''(x).$$

The assertion that the new FCS can be correctly generated by modifying the existing FCS as shown is equivalent to claiming that the basic CRC algorithm is a linear operation. That is to say that given two input texts, $A(x)$ and $B(x)$, the following property holds:

$$\text{CRC}[A(x) + B(x)] = \text{CRC}[A(x)] + \text{CRC}[B(x)]$$
$$= a(x) + b(x),$$

where CRC $(\cdot)$ is the FCS of the argument polynomial $(\cdot)$. Thus,

$$\text{CRC}[A(x)] = a(x),$$

and

$$\text{CRC}[B(x)] = b(x).$$

The claim of linearity is shown to hold by Berlekamp [3] in his discussion of the properties of congruence, as well as by Peterson [4], and will not be duplicated here.

3. Modifying the ANSI standard 32-bit frame-check sequence

• *A more complex CRC*

Many communication systems of current interest, such as the IEEE 802-series networks, employ the ANSI standard 32-bit CRC algorithm. A specification of this algorithm may be found in the 1985 IEEE 802.5 standard [5]. This particular CRC specifies several steps beyond simply dividing the text by the generating polynomial. These added steps result in a more complex algorithm for modifying the FCS in response to altered text, as they corrupt the linearity exhibited by the basic CRC algorithm.

The frame-check sequence provided by the ANSI CRC is the one's complement of the sum (modulo 2) of the following:

- The remainder of $(x^k)\,(x^{31} + x^{30} + x^{29} + \cdots + x + 1)$ divided (modulo 2) by the generating polynomial $G(c)$, where k is the number of bits in $I(x)$ (the text to be protected), and $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ is the generating polynomial.
- The remainder after multiplication (modulo 2) by x^{32} and then division (modulo 2) by $G(x)$ of the text to be protected, $I(x)$.

• *An algorithm for modifying the FCS*

Let $P(x)$ be the polynomial whose nonzero terms correspond to the bits of $(x^{32})I(x)$ that are to be altered. To modify the FCS generated by the ANSI standard CRC, first form the dummy polynomial $D(x)$:

$$D(x) = (x^k)L(x) = P(x),$$

where

$$L(x) = x^{31} + x^{30} + x^{29} + \cdots + x + 1,$$

and k is the number of bits in the text to be protected.

Let MASK* be the FCS resulting from computing the standard CRC for $D(x)$. Finally, let MASK be the one's-complement of MASK*. The modified FCS corresponding to the altered text is then

$$\text{FCS2} = \text{FCS1} + \text{MASK},$$

where FCS2 is the FCS of the altered text, FCS1 is the FCS of the original text, and MASK is the construction described above.

• *Validity of the algorithm*

The algorithm presented for modifying the FCS generated by the ANSI standard 32-bit CRC can be stated as a theorem and rigorously proved. To accomplish this, the ANSI standard algorithm is first described in mathematical terms. The theorem is then stated.

Throughout this discussion the following terminology and notation are used:

- REM$(\cdot)$ indicates the remainder after dividing the argument polynomial $(\cdot)$ by the ANSI standard generating polynomial $G(x)$.
- The operation of complementing a binary coefficient or a polynomial (by complementing each of its coefficients) is denoted by *; thus, $A(x)^*$ indicates the complement of $A(x)$.
- $I(x)$ is the text to be protected by the CRC. Stated formally, the FCS is then given by

$$\text{FCS} = [\text{REM}\,(x^k)L(x) + \text{REM}(x^{32})I(x)]^*.$$

- FCS1 is the frame-check sequence of the unmodified text. Stated formally,

$$\text{FCS1} = \{\text{REM}[(x^k)L(x)] + \text{REM}[(x^{32})I(x)]\}^*.$$

- $P(x)$ is the polynomial whose nonzero coefficients correspond to the bits of $(x^{32})I(x)$ that are to be modified.
- FCS2 is the frame-check sequence of the modified text. Stated formally,

$$\text{FCS2} = \{\text{REM}[(x^k)I(x)] + \text{REM}[(x^{32})I(x) + P(x)]\}^*.$$

- MASK is the polynomial described earlier. Stated formally,

$$\text{MASK} = (\{\text{REM}[(x^k)L(x)] + \text{REM}[(x^k)L(x) + P(x)]\}^*)^*.$$

As is shown in the proof, MASK may be simplified to

$$\text{MASK} = \text{REM}\ P(x).$$

The more complex representation of MASK is preserved here, since intermediate results such as REM $P(x)$ may not be accessible to an implementer working with an existing CRC subroutine or microchip.

Theorem Given the three conditions

1. $\text{FCS1} = \{\text{REM}[(x^k)L(x)] + \text{REM}[(x^{32})I(x)]\}^*$,
2. $\text{FCS2} = \{\text{REM}[(x^k)L(x)] + \text{REM}[(x^{32})I(x) + P(x)]\}^*$,
3. $\text{MASK} = (\{\text{REM}[(x^k)L(x)] + \text{REM}[(x^k)L(x) + P(x)]\}^*)^*$,

then

$$\text{FCS2} = \text{FCS1} + \text{MASK}.$$

Proof Using the linearity property of REM (described in the discussion of congruence), and removing the double application of complement, MASK may be simplified:

$$\begin{aligned}\text{MASK} &= (\{\text{REM}[(x^k)L(x)] + \text{REM}[(x^k)L(x) + P(x)]\}^*)^* \\ &= \{\text{REM}[(x^k)L(x)] + \text{REM}[(x^k)L(x) + P(x)]\} \\ &= \text{REM}[(x^k)L(x)] + \text{REM}[(x^k)L(x) + P(x)] \\ &= \text{REM}(x^k)L(x) + \text{REM}(x^k)L(x) + \text{REM}P(x) \\ &= \text{REM}P(x).\end{aligned}$$

Now, using the simplified version of MASK,

$$\begin{aligned}\text{FCS1} + \text{MASK} &= \{\text{REM}[(x)L(x^k)] \\ &\quad + \text{REM}[(x^{32})I(x)]\}^* + \text{REM}P(x) \\ &= P_1(x)^* + P_2(x).\end{aligned}$$

Using Lemma 2 (see below):

$$\begin{aligned}\text{FCS1} + \text{MASK} &= [P_1(x) + P_2(x)]^* \\ &= \{\text{REM}(x^k)L(x)] \\ &\quad + \text{REM}[(x^{32})I(x)] + \text{REM}P(x)\}^* \\ &= \{\text{REM}[(x^k)L(x)] \\ &\quad + \text{REM}[(x^{32})I(x) + P(x)]\}^* \\ &= \text{FCS2}.\end{aligned}$$

In the following lemmas, F is the Galois field having elements 0 and 1 from which the polynomials' coefficients are drawn. The symbol * is used to indicate the complement of the associated expression.

Lemma 1
If a and b are members of F, then

$$a + b^* = (a + b)^*.$$

Proof This lemma is easily established by an exhaustive examination of the following cases:

$$0 + 0^* = 0 + 1 = 1 = 0^* = (0 + 0)^*,$$

$$0 + 1^* = 0 + 0 = 0 = 1^* = (0 + 1)^*,$$

$$1 + 0^* = 1 + 1 = 0 = 1^* = (1 + 0)^*,$$

$$1 + 1^* = 1 + 0 = 1 = 0^* = (1 + 1)^*.$$

Lemma 2 If $P_1(x)$ and $P_2(x)$ are polynomials of degree N with coefficients drawn from F, then

$$P_1(x) + P_2(x)^* = [P_1(x) + P_2(x)]^*.$$

Proof Let $a(i)$ and $b(i)$ represent the coefficients of $P_1(x)$ and $P_2(x)$, respectively:

$$P_1(x) = \sum_{i=1}^{N} a(i)x^i,$$

$$P_2(x) = \sum_{i=1}^{N} b(i)x^i,$$

$$P_2(x)^* = \sum_{i=1}^{N} [b(i)^*]x^i,$$

$$P_1(x) + P_2(x)^* = \sum_{i=1}^{N} [a(i) + b(i)^*]x^i;$$

now, using Lemma 1,

$$P_1(x) + P_2(x)^* = \sum_{i=1}^{N} \{[a(i) + b(i)]^*\}x^i = [P_1(x) + P_2(x)]^*.$$

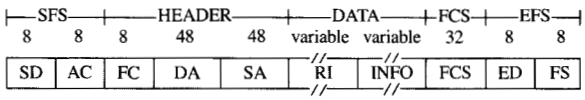
4. An example: Bridging FDDI and token-ring networks

- *The networks and the problem*

FDDI networks are described by the draft standards issued by the ANSI X3T9.5 committee [6–9]; these networks support 100-megabit-per-second transmission via optical fibers. Token-ring networks are described by the IEEE 802.5 standard [5]. Both kinds of networks are configured as rings, with medium access (permission to transmit) controlled by a token passed around the ring. Some straightforward descriptions of these networks are given in [10–14].

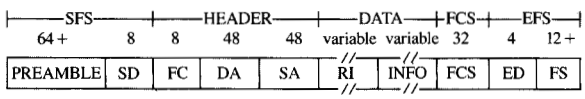
Given the high transmission rate of the FDDI network, it may serve well as a "backbone" for the lower-speed 802.5 token-ring network. This means that the FDDI network would serve as an intermediary through which messages would flow, originating on one token ring, destined for another token ring, passing through the FDDI backbone shared by many attached token rings. Providing connectivity between the token rings and the FDDI network is known as "bridging."

802.5 (Token Ring)
Numbers indicate field length in bits



SD = Starting Delimiter
AC = Access Control (PCF-0)
FC = Frame Control (PCF-1)
DA = Destination Address
SA = Source Address
SFS = Starting Frame Sequence
RI = Routing Information
INFO = Information
FCS = Frame Check Sequence
ED = Ending Delimiter
FS = Frame Status (PCF-E)
EFS = Ending Frame Sequence

FDDI
Numbers indicate field length in bits



SD = Starting Delimiter
FC = Frame Control
DA = Destination Address
SA = Source Address
SFS = Starting Frame Sequence
RI = Routing Information
INFO = Information
FCS = Frame Check Sequence
ED = Ending Delimiter
FS = Frame Status
EFS = Ending Frame Sequence

Figure 1

Comparison of token-ring and FDDI frame formats. FCS coverage extends from the FC to the FCS fields in both cases, but the contents of the two FC fields will differ.

Network	*FC format*	*Meaning*
802.5	FF ZZZ ZZZ	
	01	indicates LLC frame
	000	reserved
	YYY	LLC priority
FDDI	CL FF Z ZZZ	
	0	asynchronous frame
	1	48-bit address
	01	indicates LLC frame
	0	reserved
	YYY	LLC priority

Figure 2

Mapping the frame control fields for 802.5 and FDDI networks: Token-ring frames that are candidates for bridging have FC = 01000YYY; FDDI frames that are candidates for bridging have FC = 01010YYY.

Both the FDDI and the 802.5 standards specify that each frame is protected by the 32-bit ANSI CRC. Because of differences in the frame headers specified by the two standards, the frame-check sequence (FCS) specified by the CRC must change as a frame traverses a bridge between a token ring and the FDDI backbone. This discussion subsequently shows how the technique proposed earlier may be applied to the bridging problem. First, the FDDI and token-ring frame structures are reviewed. Details of the proposed technique are then described.

- *Frame structures*

Figure 1 shows a comparison of the FDDI and token-ring frames. The frame-control field must be altered by the bridge. All frames bridged between the token ring and the FDDI network are assumed in this discussion to be asynchronous LLC frames. Under these conditions, the FC field for the 802.5 frames is 01000YYY; the FC field for the FDDI frames is 01010YYY (YYY indicates the LLC priority). On entering the FDDI from the 802.5 network, FC is changed to 01010YYY from 01000YYY. On entering the 802.5 network from the FDDI, FC will be changed to 01000YYY from 01010YYY. **Figure 2** illustrates the meaning of these bits.

- *Modifying the frame-check sequence*

The token-ring and FDDI standards specify the same ANSI-defined 32-bit generating polynomial for the CRC. The fields of coverage are the same. Unfortunately, the frame control fields will have different bit patterns. Consequently, the FCS will change when crossing the bridge.

Following is the algorithm for changing FCS when bridging from token-ring to FDDI:

- Let INPUT represent the incoming token-ring bit stream (SD, AC, FC, DA, SA, RI, INFO, FCS(TR), ED, FS). The sequence (FC, DA, SA, RI, INFO) is of length L in bits. FCS(TR) represents the frame-check sequence for this bit string of length L, as computed by the 32-bit ANSI algorithm described earlier.
- Form the dummy bit stream of length L (in bits) given by EFFFFFF0000$\cdots$00 (for convenience shown here in hex notation). Compute the 32-bit FCS for the dummy using the 32-bit CRC algorithm. Let the 32-bit result be known as MASK*.
- Let MASK be the one's complement of MASK*.
- Let OUTPUT represent the bits departing the bridge to enter the ANSI network (PRE, SD, FC′, DA, SA, RI, INFO, FCS(FDDI), ED, FS), where FC′ is the frame control field modified as described earlier, and FCS(FDDI) represents the FCS for the sequence (FC′, DA, SA, RI, INFO). FCS(FDDI) is computed by

FCS(FDDI) = FCS(TR) + MASK,

where + represents modulo-2 addition (exclusive-OR of the bit patterns).

Figure 3 shows the transformation from token-ring to FDDI frame format.

• *Multiple-bit changes*

Under certain conditions, the bridge may be required to alter more than the single bit in the FC field. For example, the LLC priority bits carried in the FC (given as YYY in Figure 2) may not be recognized by some implementations of the 802.5 standard. If the bridge is responsible for managing these bits, the technique represented here may be extended to include modification of these (or any other) bits. This flexibility allows further application of the technique presented here to a larger class of networking problems meeting the following constraints:

- The two networks bridged both employ the same CRC algorithm.
- The protected fields are of the same length in both networks.

The extended technique follows the same algorithm given for the single-bit technique, except that the rules for forming the dummy are more elaborate. Given the necessity to alter bits in positions $N_1, N_2, \cdots, N_k$ (position 1 is at the left-hand side of the bit streams shown in Figure 3),

- Form a dummy of length L consisting of all zeros.
- Change the bits in the dummy in positions $N_1, N_2, \cdots, N_k$ from zero to one (**Figure 4**, stage 1).
- Invert the first 32 bits of the resulting dummy (Figure 4, stage 2).
- Continue as previously described, computing the 32-bit FCS for the dummy.

Figure 4 shows an example of the generalized technique.

5. Bridging token-bus and token-ring or FDDI networks

• A more difficult problem

Token-bus local-area networks are described by IEEE Standard 802.4 [15]. Frame formats for the token-bus network are much like those of the token-ring and FDDI networks, but an important difference occurs in the transmission of the information field: The order of bit transmission is reversed within each octet of the

Incoming token-ring frame (SD, AC, ED, and FS not shown):

```
41  330234513502  60003121020D  90003451350260003121  BD6516A1
FC       DA            SA            RI + INFO         FCS (TR)

Length L = 184 bits (shown above as 46 hex characters)
           not including FCS (TR)
```

Dummy frame

```
EFFFFFFF000000000000000000000000000000000000000000

Length = 184 bits (46 hex characters)
```

Masks

```
MASK* = 9B3D62FF     (32-bit FCS for Dummy)
MASK  = 64C29D00     (one's complement of MASK*)

FCS (FDDI) = FCS (TR) + MASK      + indicates modulo-2 addition
   FCS(TR)   = BD6516A1
   MASK      = 64C29D00

   FCS(FDDI) = D9A78BA1
```

Departing FDDI frame:

```
51  330234513502  60003121020D  00003451350260003121  D9A78BA1
FC'      DA            SA            RI + INFO         FCS (FDDI)
```

Figure 3

Example showing token-ring to FDDI conversion. Modifying the FCS as shown preserves end-to-end protection in bridged networks.

information field in the token-bus network with respect to the order of transmission in the other two types of networks. Nevertheless, the frame-check sequence may be modified, rather than recalculated, by a bridge interconnecting the two types of networks. Differences in the headers are treated as described earlier for an FDDI-to-token-ring bridge, and are not considered here.

• *Accommodating the reversed octets*

As a frame crosses the bus-to-ring bridge, the order of bits within each octet must be reversed. Since the protected text is altered, the frame-check sequence must be modified. The central task in developing an algorithm to modify the FCS in this situation is finding that polynomial $P(x)$ which, when added to the protected information field, in fact reverses the bit order within octets of that field.

Incoming token-ring frame (SD, AC, ED, and FS not shown):

```
41  330234513502  60003121020D  00003451350260003121  BD6516A1
FC      DA            SA              RI + INFO         FCS (TR)

Length L = 184 bits (shown above as 46 hex characters)
           not including FCS (TR)
```

Dummy frame for altering the fourth and eighth bits

```
Evolution of dummy (in binary):
00010001000000000000000000000000000...0000 (stage 1)
11101110111111111111111111111110000...0000 (stage 2)

EEFFFFFF000000000000000000000000000000000000000000  (hex)

Length = 184 bits (46 hex characters)
```

Masks

```
MASK* = 9D714B2F   (32-bit FCS for Dummy)
MASK  = 628EB4D0   (one's complement of MASK*)

FCS (FDDI) = FCS (TR) + MASK      + indicates modulo-2 addition
    FCS(TR)   = BD6516A1
    MASK      = 628EB4D0

    FCS(FDDI) = DFEBA271
```

Departing FDDI frame:

```
50  330234513502  60003121020D  00003451350260003121  DFEBA271
FC'     DA            SA              RI + INFO         FCS (FDDI)
```

Figure 4

Example showing the change of two bits by the bridge. Bits in positions four and eight are changed; the FCS is modified accordingly.

The polynomial $P(x)$ may be assembled on an octet-by-octet basis. Each octet to be reversed serves as an address to two pre-established tables. One table contains bit patterns (polynomials) that when added to the address result in a sum having the bits of the address in reverse order. These patterns are collected sequentially, octet by octet, to construct $P(x)$. The other table contains the bits of the address in reversed order. The contents of this table are used to construct the departing information field. **Figure 5** shows an example illustrating these ideas for a four-octet information field.

To complete the algorithm, MASK is generated from $P(x)$ as previously described, and added to the original FCS as a frame traverses the bridge. Using the method described allows end-to-end CRC protection across the bus-to-ring bridge.

6. Determining the need for end-to-end CRC integrity

Undetectable errors occur naturally whenever CRCs are used to protect data sent over noisy channels. The system designer hopes to keep the frequency of occurrence of such errors to an absolute minimum, and selects a CRC with a generating polynomial appropriate to the performance objectives at hand. The rate of naturally occurring undetected errors, as opposed to errors inadvertently introduced when changing protected text as described earlier, is now estimated for systems protected by the 32-bit ANSI CRC algorithm. The result of this estimate is used to argue that the internal bit-error ratio of a machine that changes CRC-protected data must be quite tightly controlled in the absence of the technique described earlier in this paper.

According to Witzke and Leung [16], the conditional probability of undetected errors for corrupted text protected by a 32-bit CRC approaches 2^{-32} as the length of the text increases. On the basis of trends presented in [16] for 12- and 16-bit CRCs, it is assumed here that the conditional probability of undetected error for a 32-bit CRC protecting a corrupted 2000-octet text is close to the asymptotic value of 2^{-32}, or approximately 10^{-10}. The total probability of an undetected error is given by the product of the probability that errors have been introduced into a frame and the conditional probability that the error will be undetected (10^{-10}).

The probability that any given frame will contain transmission errors is not easy to establish with a great deal of precision. In the following discussion, a burst-generating error mechanism is assumed, as it can easily be shown that a burst mechanism is much more likely than a random-error mechanism to generate error patterns that are not detected by cyclic redundancy checks.

A frequency of occurrence of one burst per minute is often used in telecommunications analysis, and is used here. During an error burst, the communications channel exhibits the properties of a binary symmetric channel with crossover probability of 0.5. Assuming a burst length of 1.0 ms gives a frame-error probability between 1.9×10^{-5} (at 100 Mbps) and 6.7×10^{-5} (at 4 Mbps) for 2000-octet frames. The greater of these values leads to a total probability of undetected frame error of $10^{-10} \times (6.7 \times 10^{-5})$, or 6.7×10^{-15}. From these results, the probability of an undetected frame error native to the network is now assumed, for purposes of argument, to be typically of the order of 10^{-14}.

Suppose that it is desired to limit the rate of undetectable errors introduced by a text-changing device

to ten percent of the rate of undetected errors native to the network protected by the 32-bit CRC. This would require that the probability of the device's inserting one or more errors into a frame be less than 10^{-15}. The text-changing device is assumed to be characterized by uncorrelated, randomly occurring, internal bit errors with bit-error ratio b. Then, the probability that a frame error (one or more bit errors) will be introduced into a 2000-octet frame by the text-changing device is given by

$$\begin{aligned}\text{Prob(frame error)} &= \text{Prob(1 or more bit errors)} \\ &= 1.0 - \text{Prob(no bit errors)} \\ &= 1.0 - (1.0 - b)^{16000}.\end{aligned}$$

Using the first two terms of the binomial expansion of $(1.0 - b)^{16000}$ gives

$$\text{Prob(frame error)} = 16000 \times b.$$

Thus, an internal bit-error ratio of 6×10^{-20} is needed to support a frame-error ratio of 10^{-14}, under the assumption of randomly occurring internal errors.

The above argument, although most assuredly inexact, leads to an interesting point: If the CRC is to provide end-to-end integrity with the degree of certainty implied by the choice of a degree-32 generating polynomial, then there is a good possibility that the technique presented here for modifying the FCS in response to text changes will be beneficial for devices that include an unprotected signal path. Testing a machine to ensure, with statistical significance, that its internal bit-error ratio is less than 6×10^{-20} may be quite difficult. Any greater internal error ratio begins to erode the end-to-end system integrity. On the other hand, the degree of protection provided by a 32-bit CRC may be a case of overkill for some applications, where such careful control of undetected errors may not be required. Further, the inclusion of parity or other hardware protection in the text-changing device would significantly lessen the need for the technique proposed here.

Address (input)	$P(x)$	Reversal (output)
00000000	00000000	00000000
00000001	10000001	10000000
00000010	01000010	01000000
.	.	.
.	.	.
.	.	.
00011001	10000001	10011000
.	.	.
.	.	.
.	.	.
01011111	10100101	11111010
.	.	.
.	.	.
.	.	.
10010110	11111111	01101001
.	.	.
.	.	.
.	.	.
11011011	00000000	11011011
.	.	.
.	.	.
.	.	.
11111110	10000001	01111111
11111111	00000000	11111111

Example:

Input text	11011011	00011001	10010110	01011111
+ $P(x)$	00000000	10000001	11111111	10100101
=				
Output text	11011011	10011000	01101001	11111010

Figure 5

Constructing $P(x)$ and reversing the contents of the octets. The output text is found by table lookup, but it could be generated by adding $P(x)$ to the input text. $P(x)$ is used in generating the MASK that modifies the frame-check sequence in response to reversing the transmission order within the input's octets.

7. Summary and concluding remarks

An algorithm has been presented for preserving the end-to-end integrity of cyclic redundancy checks in networks that intentionally alter protected bits while messages are in transit. This algorithm is an application of a known mathematical property of the basic CRC—the CRC operation is linear, and, consequently, frame-check sequences may be manipulated by superposition as the protected text is altered. A rigorous proof of the suggested algorithm applicable to industry-standard CRCs has been presented. An application of this technique to an FDDI-to-token-ring bridge has been discussed in detail, and an application of the technique to the challenging problem of interconnecting token-bus and token-ring or FDDI networks has been outlined. The technique is shown to be beneficial for LAN bridges with internal bit-error ratios exceeding 10^{-19} on unprotected signal paths.

Acknowledgment

The author thanks Kian-Bon Sy, of IBM at Research Triangle Park, for his help in identifying the problem of undetectable errors, and for showing the need for a way of modifying, rather than recalculating, a frame-check sequence.

References and note

1. See for example *IEEE Draft Standard 802.1: Part D, MacBridges, Revision C* (Institute of Electrical and Electronics Engineers, 345 E. 47th St., New York, NY, August 1987), p. 20: "It is therefore necessary to recalculate the FCS within a bridge providing a relay function between 802 MACs of dissimilar types. This introduces the possibility of additional undetected bit errors." Although somewhat complicated by the reversal of the transmission order of the bits within octets of the information

field, an 802.4-to-802.5 bridge can be constructed using the techniques described here. This application is outlined in Section 5 of this paper. Bridges connecting the 802.3 to other networks, however, appear to be beyond the reach of the method proposed here.

2. S. Lin, *An Introduction to Error-Correcting Codes,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1970, Ch. 2.
3. E. R. Berlekamp, *Algebraic Coding Theory,* McGraw-Hill Book Co., Inc., New York, 1968, p. 14.
4. W. W. Peterson and D. T. Brown, "Cyclic Codes for Error Detection," *Proceedings of the IRE,* pp. 228–235 (January 1961).
5. *Token Ring Access Method and Physical Layer Specifications,* ANSI/IEEE Standard 802.5-1985, Institute of Electrical and Electronics Engineers, 345 E. 47th St., New York, 1985.
6. *FDDI Physical Layer Medium Dependent (PMD) Standard,* draft proposed American National Standard, X3T9/86-71 X3T9.5/84-48 Rev. 6.2, American National Standards Institute, Washington, DC, January 10, 1987.
7. *FDDI Physical Layer Protocol (PHY) Standard,* draft proposed American National Standard, X3T9/85-39 X3T9.5/83-15 Rev. 14, American National Standards Institute, Washington, DC, October 20, 1986.
8. *FDDI Token Ring Media Access Control (MAC) Standard,* X3T9/84-100 X3T9.5/83-16 Rev. 10, American National Standards Institute, Washington, DC, February 28, 1986.
9. *FDDI Station Management (SMT) Standard,* draft proposed American National Standard, X3T9/85-X3T9.5/84-49 Rev. 2, American National Standards Institute, Washington, DC, September 5, 1986.
10. W. E. Burr, "The FDDI Optical Data Link," *IEEE Commun. Mag.* **24,** 10–17 (May 1986).
11. F. E. Ross, "FDDI—A Tutorial," *IEEE Commun. Mag.* **24,** 18–23 (May 1986).
12. V. Iyer and S. Joshi, "FDDI's 100M-bps Protocol Improves on 802.5 Spec's 4M-bps Limit," *Electronic Design News,* pp. 151–160 (May 2, 1985).
13. Susan Wallach, "FDDI Tutorial," *LAN Mag.,* pp. 44–47 (March 1987).
14. Norman C. Strole, "A Local Communications Network Based on Interconnected Token-Access Rings: A Tutorial," *IBM J. Res. Develop.* **27,** 481–496 (September 1983).
15. *Token-Passing Bus Access Method,* ANSI/IEEE Standard 802.4-1985, Institute of Electrical and Electronics Engineers, 345 E. 47th St., New York, 1985.
16. K. A. Witzke and C. Leung, "A Comparison of Some Error-Detecting CRC Code Standards," *IEEE Trans. Commun.* **COM-33,** 996–998 (September 1985).

Received December 16, 1988; accepted for publication November 21, 1989

David R. Irvin *IBM Communication Systems, U.S. Telecommunications Center, P.O. Box 12195, Research Triangle Park, North Carolina 27709.* Mr. Irvin is an advisory engineer/scientist at the IBM U.S. Telecommunications Center at Research Triangle Park. Since joining IBM in 1974, he has worked in the fields of telecommunication analysis, communications and network management architecture, system performance analysis, transmission technology, and digital signal processing. Prior to joining IBM, he worked under contract for the United States Navy. Mr. Irvin received a B.E.S. in 1970 from The Johns Hopkins University at Baltimore, and the M.E.E. in 1971 from North Carolina State University at Raleigh. He is a member of Phi Beta Kappa, Tau Beta Pi, and Eta Kappa Nu, and a Senior Member of the Institute of Electrical and Electronics Engineers.