SANDIA REPORT

SAND91-1004 • UC-705

Unlimited Release

Printed June 1991

6.28.91

Authentication of Byte Sequences

Samuel D. Stearns

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
US Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND--91-1004

DE92 000150

SAND91-1004
Unlimited Release
Printed June 1991

AUTHENTICATION OF BYTE SEQUENCES

Samuel D. Stearns
Ground Motion & Seismic Division 9311
Sandia National Laboratories
Albuquerque, NM 87185

ABSTRACT

Algorithms for the authentication of byte sequences are described. The algorithms are designed to authenticate data in the Storage, Retrieval, Analysis, and Display (SRAD) Test Data Archive of the Radiation Effects and Testing Directorate (9100) at Sandia National Laboratories, and may be used in similar situations where authentication of stored data is required.

The algorithms use a well-known error detection method called the Cyclic Redundancy Check (CRC). When a byte sequence is authenticated and stored, CRC bytes are generated and attached to the end of the sequence. When the authenticated data is retrieved, the authentication check consists of processing the entire sequence, including the CRC bytes, and checking for a remainder of zero. The error detection properties of the CRC are extensive and result in a reliable authentication of SRAD data.

ACKNOWLEDGEMENTS

The author wishes to thank Frank Biggs (9312), Lynn Kissel (9312), and Tom Marking (9312) who have provided essential guidance and testing during the development of the algorithms described in this report.

TABLE OF CONTENTS

1. Introduction.....	7
2. Fortran Routines.....	7
3. Examples.....	9
4. Cyclic Codes.....	10
5. Error Detection Capabilities.....	13

FIGURE

Fig. 1. Calling sequences for ENCOD and DECOD routines.....	8
---	---

TABLE

Table 1. Error Detection Properties.....	13
--	----

1. Introduction

This report describes algorithms used to authenticate data in the Storage, Retrieval, Analysis, and Display (SRAD) archive system [1] of the Radiation Effects and Testing Directorate (9100) at Sandia Labs. Since most of this data is field test instrumentation waveform data recorded at the Nevada Test Site, the motivation for such authentication is obvious. The tests that produce the data are expensive, complex, and often cannot be repeated; therefore the data is irreplaceable and must be made capable of self-authentication. Some authentication is provided by error detection in the storage hardware (various tape and disk drives), but with this data, further authentication is needed.

The authentication algorithms operate on SRAD data in the form of byte sequences. Typically, the SRAD data is the output byte sequence from one of the data compression routines [1,2]. The algorithms themselves exist in the form of Fortran-77 routines, and are designed to be portable, that is, the routines run on a PC as well as a VAX.

The algorithms produce a Cyclic Redundancy Check (CRC) which is one to four bytes long and is appended to an SRAD sequence before the latter is stored. When the sequence is retrieved from storage, the algorithms process the entire sequence, including the CRC bytes, to check for errors.

The CRC, also known as a "cyclic code" [3,4] is used because of its powerful error detecting properties, particularly with burst errors where a localized area of the storage medium is affected. As seen in section 5, these properties lead to near-100% reliability of the retrieved data.

2. Fortran Routines

The authentication algorithms are incorporated into four Fortran subroutines. One pair of routines, ENCODC and DECODC, is used to encode and decode "short" byte sequences that can be placed, all at once, in a Fortran character array. The definition of "short" depends on the computer -- a PC may only allow an array size on the order of 10^4 , whereas a VAX typically allows array sizes on the order of 10^7 . These routines use from one to four CRC bytes in each authenticated sequence.

The second pair of routines, ENCODL AND DECODL, is used to encode and decode "long" byte sequence that cannot fit all at once into a Fortran character array. These routines are usually called repeatedly and process the byte sequence in pieces. They always use four CRC bytes in each authenticated sequence.

The calling sequences for all four routines are given in Fig. 1. The first routine, ENCODC, is used in the following way: We place a sequence of N_c bytes into a Fortran "character*1" array, $ic(0:N_c-1)$. Then we call ENCODC, which produces N_b (from one to four) CRC bytes. Finally, we append the CRC bytes to the original sequence to produce an authenticated byte sequence with $N_d=N_c+N_b$ bytes.

The second routine, DECODC, operates on the authenticated byte sequence with $N_d=N_c+N_b$ bytes produced using ENCODC. Its only output is

an error indicator, *ierror*, that indicates whether any of the *Nd* bytes have been altered in a detectable manner.

The other two routines, ENCODL and DECODL, operate together in a similar manner. Normally, ENCODL is called first with *Npass*=1, then with *Npass*=2 as many times as needed, and finally with *Npass*=3. Four CRC bytes are produced by ENCODL on the final pass with *Npass*=3. Then, to check authentication, DECODL is called in the same way and the error indicator, *ierror*, is checked after the final pass. (In the final pass with DECODL, *ic*(0:*Nd*-1) includes the four CRC bytes.)

Call ENCODC(<i>ic</i> , <i>Nc</i> , <i>Nb</i> , <i>icheck</i>)		
Call DECODC(<i>ic</i> , <i>Nd</i> , <i>Nb</i> , <i>ierror</i>)		
Call ENCODL(<i>ic</i> , <i>Nc</i> , <i>Npass</i> , <i>icheck</i>)		
Call DECODL(<i>ic</i> , <i>Nd</i> , <i>Npass</i> , <i>ierror</i>)		
<i>ic</i>	= input byte array	(character*1)
<i>Nc</i> , <i>Nd</i>	= # bytes in <i>ic</i> In DECODC, <i>Nd</i> = <i>Nc</i> + <i>Nb</i> .	(integer*4)
<i>Nb</i>	= # CRC bytes (1-4)	(integer*4)
<i>icheck</i> (<i>Nb</i>)	= CRC byte(s) produced by ENCOD routines. (<i>Nb</i> =4 in ENCODL.)	(character*1)
<i>ierror</i>	= 0: no errors. 1: error detected by a DECOD routine	(integer*4)
<i>Npass</i>	= 0: entire sequence is in <i>ic</i> 1: sequence begins with <i>ic</i> 2: sequence continues with <i>ic</i> 3: sequence ends with <i>ic</i>	(integer*4)

Fig. 1. Calling sequences for ENCOD and DECOD routines.

The "*ic*" array containing the input byte array is dimensioned as follows:

```
in ENCOD routines:  ic(0:Nc-1)
in DECOD routines:  ic(0:Nd-1).
```

Also, note that calling ENCODL with *Npass*=0 is equivalent to calling ENCODC with *Nb*=4. Since ENCODC is simpler to use and allows a choice of *Nb*, the latter is usually preferable if ENCODC is available.

3. Examples

Two examples are presented here. The first is an example using ENCODC and DECODC in Program CRCEX1, which is shown below. The input byte sequence is initially in the character string labeled "input" and in this case is defined in the DATA statement as a sequence of printable characters. Most compressed SRAD data consists of nonprintable bytes that represent numerical data. In any case, printability of the ic array is not a requirement of the routines.

```
Program CRCEX1
C-CRC Example 1 -- using ENCODC and DECODC.
C-
  Character*18 input
  Character*1 ic(0:19)
  Data input/"In the beginning.."/
  read(input,'(18a1)') (ic(j),j=0,17)
  call ENCODC(ic,18,2,ic(18))
  print '(/1x,18a3/1x,20(1x,z2))', (ic(j),j=0,17), (ic(j),j=0,19)
  call DECODC(ic,20,2,ie1)
  ic(7)=char(ichar(ic(7))+1)
  call DECODC(ic,20,2,ie2)
  print '(/'' ie1,ie2:''',2i2)', ie1,ie2
end

  I n   t h e   b e g i n n i n g   .   .
49 6E 20 74 68 65 20 62 65 67 69 6E 6E 69 6E 67 2E 2E A5 5E

ie1,ie2: 0 1
```

The input sequence is placed in the ic array by the READ statement. The results of ENCODC with Nb=2 check bytes are printed, first alphabetically and then in hexadecimal. The two CRC bytes are also printed in hexadecimal (A5,5E) beyond the last input byte. Note that in this example ENCODC places the CRC bytes at the end of the ic array in ic(18) and ic(19).

Next, DECODC is called to check the authenticity of the complete ic sequence, now of length Nd=18+2=20. The indicator, ie1, indicates successful authentication with no errors detected.

Finally, a single bit of the authenticated sequence is switched by adding one to ic(7) (changing "b" to "c"), and DECODC is again called. The result in ie2 indicates that the error was detected.

The second example uses ENCODL and DECODL in Program CRCEX2, shown below. The input sequence in "input" is too long to be read all at once into the ic array, and so ENCODL is called three times with Npass equal

1, 2, and 3. After the third pass, the four icheck bytes are printed in hexadecimal and placed at the end of the input sequence. Finally, the entire input sequence is read into the id array, and DECODL is called with Npass equal 0 to check the sequence in a single pass. The error indicator ie=0 indicates a valid authentication.

```

      Program CRCEX2
C-CRC Example 2 -- using ENCODL and DECODL.
C-
      Character*22 input
      Character*1 ic(0:5),icheck(4),id(22)
      Data input/"In the beginning..    "/"
      read(input(1:6),'(6a1)') ic
      call ENCODL(ic,6,1,icheck)
      read(input(7:12),'(6a1)') ic
      call ENCODL(ic,6,2,icheck)
      read(input(13:18),'(6a1)') ic
      call ENCODL(ic,6,3,icheck)
      print '(/' icheck:''',4(1x,z2))', icheck
      do 1 j=1,4
1   input(18+j:18+j)=icheck(j)
      read(input,'(22a1)') id
      call DECODL(id,22,0,ie)
      print '('  ie:''',i2)', ie
      end

      icheck: 4D CE AA F0
      ie: 0

```

4. Cyclic Codes

Cyclic codes [3,4] are basically binary codes, in which a binary input sequence is processed to produce a sequence of CRC bits. The latter are appended to the end of the input sequence to produce the authenticated sequence, just as in the preceding examples. Then the entire sequence (with possible alterations) is processed in the same way, this time to check authenticity.

The processing is equivalent to mod-2 division, in which an input polynomial, $P(x)$, is divided (mod-2) by a generating polynomial, $G(x)$, to produce a quotient, $Q(x)$, and a remainder, $R(x)$. Let the binary input sequence be

$$\text{binary input sequence: } i_0, i_1, \dots, i_{N-1} \quad (1)$$

These may be considered as the bits of the character array, $ic(0:Nc-1)$, eight bits to a character. The corresponding input polynomial is

$$P(x) = i_0x^{N-1} + i_1x^{N-2} + \dots + i_{N-1}x^0 \quad (2)$$

The generating polynomial is

$$G(x) = g_0x^M + g_1x^{M-1} + \dots + g_Mx^0 \quad (3)$$

The mod-2 division is described by

$$\frac{x^M P(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)} \quad (4)$$

The remainder, $R(x)$, is of degree less than M , the degree of $G(x)$. The following example, in which $N=3$, $M=3$, $(i_0, i_1, i_2) = (1, 0, 0)$, and $(g_0, \dots, g_3) = (1, 0, 1, 1)$, illustrates the mod-2 division in Eq.4. (Each mod-2 subtraction in the division process is equivalent to an exclusive-or operation.)

$$\begin{array}{r} 101 \\ 1011 \overline{) 100 \ 000} \\ \underline{101 \ 1} \\ 01 \ 10 \\ \underline{00 \ 00} \\ 1 \ 100 \\ \underline{1 \ 011} \\ 111 \end{array}$$

$$\begin{aligned} P(x) &= x^2 \\ x^M P(x) &= x^5 \\ G(x) &= x^3 + x + 1 \\ Q(x) &= x^2 + 1 \\ R(x) &= x^2 + x + 1 \end{aligned}$$

This example can be verified by proving $x^M P(x) = G(x)Q(x) + R(x)$ using mod-2 multiplication.

Now suppose we generate an authenticated sequence in the manner just described, by appending $R(x)$ to $P(x)$, to produce the authenticated sequence,

$$A(x) = x^M P(x) + R(x) \quad (5)$$

If $A(x)$ is processed as above, that is, divided by $G(x)$, then, using Eq.4, we have

$$\begin{aligned} \frac{A(x)}{G(x)} &= \frac{x^M P(x)}{G(x)} + \frac{R(x)}{G(x)} \\ &= Q(x) + \frac{R(x)}{G(x)} + \frac{R(x)}{G(x)} = Q(x) \end{aligned} \quad (6)$$

The final result follows because any function added to itself mod-2 is zero. Thus, an authentication check consists of dividing $A(x)$ by $G(x)$ and checking for a zero remainder. Using the example above with $N=M=3$, we have $A(x)=x^5+x^2+x+1$, and the division by $G(x)$ is as follows:

$$\begin{array}{r}
101 \\
1011 \overline{) 100111} \\
\underline{1011} \\
0101 \\
\underline{0000} \\
1011 \\
\underline{1011} \\
0
\end{array}$$

$$\begin{aligned}
A(x) &= x^5 + x^2 + x + 1 \\
G(x) &= x^3 + x + 1 \\
Q(x) &= x^2 + 1 \\
R(x) &= 0
\end{aligned}$$

All four of the routines described in this report use the mod-2 division process just described to produce the authentication bytes and to check the authenticated data. An authentication check in either DECOD routine amounts to a division of $A(x)$ by the same $G(x)$ used in the corresponding ENCOD routine and a check for a zero remainder.

Any alteration of an authenticated sequence is equivalent to adding an error polynomial, $E(x)$, to $A(x)$. Then, when $A(x)$ is processed as in Eq.6, we have the additional term, $E(x)/G(x)$. The alteration is detected if there is a remainder, that is, if and only if $E(x)$ is not evenly divisible by $G(x)$.

More generally, a "burst" of errors may be represented by $x^k E(x)$, with the first and last coefficients of $E(x)$ both equal to one. The length of the burst, from the first through the last error, is one plus the degree of $E(x)$. The burst is detectable if and only if $x^k E(x)$ is not evenly divisible by $G(x)$.

An *irreducible polynomial* is a polynomial that is prime, that is, has no factors other than one. In our four CRC options with $Nb=1,2,3$, and 4, $G(x)$ is the product of $(x+1)$ times $I(x)$, an irreducible polynomial of degree $M-1 = (8 \cdot Nb - 1) = 7, 15, 23$, and 31, respectively. When $G(x)$ is of this form, certain error detection capabilities result. For example, one can show easily that, if the burst length is M or less, $E(x)$ cannot be evenly divided by $G(x)$, and the error burst is always detected.

The most important error detection capabilities are summarized in the next section. These capabilities depend on each $I(x)$ being *primitive* as well as irreducible. A primitive polynomial of degree $M-1$ is an irreducible polynomial, $I(x)$, with the following property: The smallest integer, e , for which $x^e + 1$ is evenly divisible by $I(x)$, is $e=2^{M-1}-1$. The following primitive and generating polynomials, with $G(x)=(x+1)I(x)$ and with approximate values of M and e as shown, are used in our routines:

Nb=1:	$I(x) = x^7 + x^3 + 1$ $G(x) = x^8 + x^7 + x^4 + x^3 + x + 1$	$M=8$ $e=127$
Nb=2:	$I(x) = x^{15} + x + 1$ $G(x) = x^{16} + x^{15} + x^2 + 1$	$M=16$ $e=3.277E2$
Nb=3:	$I(x) = x^{23} + x^5 + 1$ $G(x) = x^{24} + x^{23} + x^6 + x^5 + x + 1$	$M=24$ $e=8.389E6$
Nb=4:	$I(x) = x^{31} + x^3 + 1$ $G(x) = x^{32} + x^{31} + x^4 + x^3 + x + 1$	$M=32$ $e=2.147E9$

A final useful property of the above polynomials is that each has a minimum number of nonzero coefficients [4]. This property results in the shortest computing time.

5. Error Detection Capabilities

When ENCODC is used to authenticate data, the error detection capabilities of DECODC depend on the choice of Nb, the number of check bytes. With ENCODL and DECODL, Nb is always four.

In any case, when SRAD data is being authenticated, the data sequences are generally long, and the most important detection property is the fraction of undetected errors. For burst lengths over 33 bits, this fraction is 2^{-M} .

A second important capability was mentioned in the preceding section: Any burst of length M or less is always detected. These and other error detection capabilities are summarized in Table 1, where values of M and e from the preceding section are used. All capabilities are derived by Ingels [3].

Table 1. Error Detection Properties

Nb = Number of CRC bytes

b = burst length in bits

Nc = sequence length in bytes

	Nb=1	Nb=2	Nb=3	Nb=4
M	8	16	24	32
e	127	3.277E2	8.389E6	2.147E9
Fraction of undetected bursts with $b \leq 8Nb$:	0.0	0.0	0.0	0.0
with $b = 8Nb+1$:	7.81E-3	3.05E-3	1.19E-7	4.66E-10
with $b > 8Nb+1$:	3.91E-3	1.53E-5	5.96E-8	2.33E-10

Additional capabilities:

- Any burst with 2 or 3 error bits detected if $8Nc \leq e$.
- All bursts with b odd detected.
- Any 2 bursts, both with $b \leq 2$ detected if $8Nc \leq e$.

References

1. Kissel, L., T.R. Marking, and F. Biggs, "UFO (UnFold Operator): Default Data Format", Sandia National Laboratories SAND91-0490, to be released.
2. Stearns, S.D., "Predictive Data Compression with Exact Recovery", Sandia National Laboratories SAND90-2583, December, 1990.
3. Ingels, F.M., Information and Coding Theory, Intext Educational Publishers, 1971, Chapter 7.
4. Peterson, W.W., and E.J. Weldon, Error-Correcting Codes, 2nd Ed., Massachusetts Institute of Technology, 1961, Appendix C.

Distribution:

Mr. Glenn Lommasson
Gulton Industries, Inc.
P.O. Box 3027
Albuquerque, NM 87190

Dr. C. R. Hutt / Ms. C. V. Peterson
Albuquerque Seismological Laboratory
Bldg. 10002, KAFB East
Albuquerque, NM 87115-5000

Prof. D. M. Etter
3167 Nelson Rd.
Longmont, CO 80503

Prof. G. W. Donohoe
EECE Dept.
University of New Mexico
Albuquerque, NM 87131

Prof. N. Ahmed
EECE Dept.
University of New Mexico
Albuquerque, NM 87131

200	G. J. Simmons
2813	J. R. Yoder
2931	S. K. Fletcher
2932	R. E. Jones
2940	W. F. Mason
2943	A. R. Iacoletti
2943	M. D. Olman
5144	D. E. Ryerson
5145	D. H. Schroeder
5220	J. L. Rea
5234	C. L. Schuster
7231	J. A. Cooper
7262	K. D. Flynn
7522	O. M. Solomon
7550	R. A. David
9112	J. R. Wayland
9130	R. D. Andreas
9212	G. R. Elliott
9240	P. A. Stokes
9241	T. S. McDonald
9243	P. B. Herrington
9243	D. R. Breeding
9243	D. O. Lee
9300	J. E. Powell
9310	J. D. Plimpton
9311	A. J. Chabai
9311	J. R. Banister
9312	K. M. Glibert
9312	F. Biggs
9312	J. D. Harmon
9312	L. D. Kissel
9312	T. Marking
9320	M. J. Navratil
9321	W. B. Boyer
9321	R. D. Aden
9321	P. C. Kaestner
9321	W. G. Perkins
9322	C. W. Cook
9322	P. L. Wehrman
9325	D. C. Browne
3141	S. A. Landenberger (5)
3145	Document Processing For DOE/OSTI
3151	G. C. Claycomb (3)
8523	R. C. Christman

END

**DATE
FILMED**

10 129191

11