This worksheet shows the polynomial math behind the iSCSI CRC32c polynomial. I now take into account bit reflection (reversing bits within a byte) and complementing remainder of the polynomial divison and initializing the CRC to 1s.
The data set that I use to test the process consists of 32 decrementing bytes which I generate using my new function IntegersToPoly. I also add, to the transmitted message, a particular error polynomial that I know will not be detected by the CRC. The addition of this error polynomial should have no effect on the results of the CRC checker portion of the algorithm.
You don't need to understand *Mathematica* syntax to understand this worksheet. Reading the comments should suffice.

my function definitions

```
IntegersToPoly[list1_, base_, bitsPerWord_] := Module[
  {list2, list3, list4, order}, (* local variables *)
  list2 = IntegerDigits[list1, base, bitsPerWord]; (* convert each digit to an 8 bit binary word *)
  list3 = Flatten[list2]; (* convert nested list to flat list *)
list4 = PadLeft[list3, 32 * (1 + Quotient[Length[list3], 32])]; (* pad with zeroes on the left so have multiple of 32 *)
order = Length[list4];
Sum[Part[list4, i] * x^(order - i), {i, 1, order}]
  (* multiply each list member with appropriate power of x and accumulate sum. Leftmost or
    first bit is the most significant coefficient and thus multiplies the highest power of x *)
 ]
```

```
IntegersToPoly::usage = "This function converts a list of integers – most significant
   on the left – into a polynomial with binary coefficients.The binary representation of the
   list is padded with 0 in most significant positions so the number of bits is a multiple of 32."
```

```
This function converts a list of integers – most significant on the
  left – into a polynomial with binary coefficients.The binary representation of the list
  is padded with 0 in most significant positions so the number of bits is a multiple of 32.
```

```
PolyToIntegers[poly1_, bitsPerWord_] := Module[
  {list1, list2, list3, list4},
  list1 = CoefficientList[poly1, x];
  (* list of binary coefficients. list starts (left) with the coefficient of the 0 th power of x *)
  list4 = Reverse[list1]; (*want most significant bit on left*)
  list2 = PadLeft[list4, 32 * (1 + Quotient[Length[list1], 32])];
  (* pad with zeroes on the left to nearest multiple of 32 *)
  list3 = Partition[list2, bitsPerWord]; (* generates sublists of "bitsPerWord" bits *)
  Map[binToDec, list3] (* interprets list in binary and converts to decimal digits *)
 ]
```

```
PolyToIntegers::usage = "groups the binary poly coefficients in groups of <bitsPerWord> and displays in decimal "
```

```
groups the binary poly coefficients in groups of <bitsPerWord> and displays in decimal
```

```
reverseByteBits[poly1_] :=
 Module[
  {list1, list2, list3, list4, list5, list6, list7, list8, list9, list3Length, list4Length}, (* local variables *)
  list2 = CoefficientList[poly1, x]; (* list starts with power 0 *)
  list3 = Reverse[list2];
  list3Length = Length[list3];
  list4 = PadLeft[list3, (Quotient[list3Length, 32] + 1) * 32]; (* pad to next higher multiple of 32 *)
  list4Length =  Length[list4];
  list5 = Partition[list4, 8]; (* generates sublists of 8 bits *)
  list6 = Map[Reverse, list5]; (* reverse bits within each sublist *)
  list7 = Flatten[list6]; (* back to single list of coefficients of the desired poly*)
  list8 = Reverse[Table[x^i, {i, 0, list4Length - 1}]]; (* create list the size of list4 of incrementing powers of x *)
  list9 = list7 * list8; (* multiply the two lists to create a list containing terms of the desired poly *)
  Apply[Plus, list9] (* add the terms to get the poly *)

 ]
```

```
reverseByteBits::usage = "transforms a  polynomial of arbitrary degree into a new polynomial with
   the order of coefficients in each group of 8 terms reversed. This has been called bit reflection"
```

```
transforms a  polynomial of arbitrary degree into a new polynomial with the
  order of coefficients in each group of 8 terms reversed. This has been called bit reflection
```

```
binToHex[inlist_] :=
 Module[{list1},
  list1 = FromDigits[inlist, 2]; (* interpret the list as digits of a binary number *)
  BaseForm[list1, 16] (* display in base 16 *)
 ]
```

```
binToHex::usage = "binToHex displays in hex its argument - a list of bits. "

binToHex displays in hex its argument - a list of bits.

binToDec[inlist_] :=
 Module[{list1},
  list1 = FromDigits[inlist, 2]; (* interpret the list as digits of a binary number *)
  BaseForm[list1, 10] (* display in base 10 *)
 ]

binToDec::usage = "binToDec displays in decimal its argument - a list of bits. "

binToDec displays in decimal its argument - a list of bits.



polyCoeffsToHex[poly1_] :=
       Module[
       {list1,list2,list3,list4}, (* the local variables *)
       list1=CoefficientList[poly1,x]; (* extracts coefficients of the poly *)
       list2=Reverse[list1]; (* reverses so higher order coeffs are on left side *)
       list3=PadLeft[list2,32]; (* pads coefficients of higher order terms if missing *)
       list4=Partition[list3,4]; (* partitions in groups of 4 *)
       Map[binToHex,list4] (* converts each group of 4 to hex . *)
       ]
```

```
polyCoeffsToHex::usage = "polyCoeffsToHex gives in hex the coefficients of an order 32 polynomial. Makes use of binToHex"
```

```
polyCoeffsToHex gives in hex the coefficients of an order 32 polynomial. Makes use of binToHex
```

```
allOnesPoly[size_] :=
 Module[
  { listOfIntegers, bitsPerWord, base }, (* local variables *)
listOfIntegers = Table[1, {size}]; (* generates a list of <size> 1 s *)
bitsPerWord = 1; (* each integer will be represented with a 1 bit binary word *)
base = 2; (* like I said, in binary *)
  F = IntegersToPoly[listOfIntegers, base, bitsPerWord]
 ]
```

```
allOnesPoly::usage = " generates a poly of <size> ones"
```

```
 generates a poly of <size> ones
```

end of my function definitions

The CRC32c polynomial used in iSCSI

```
G = x^32 + x^28 + x^27 + x^26 + x^25 + x^23 + x^22 + x^20 + x^19 + x^18 + x^14 + x^13 + x^11 + x^10 + x^9 + x^8 + x^6 + 1
```

$1 + x^6 + x^8 + x^9 + x^{10} + x^{11} + x^{13} + x^{14} + x^{18} + x^{19} + x^{20} + x^{22} + x^{23} + x^{25} + x^{26} + x^{27} + x^{28} + x^{32}$

A degree 31 constant polynomial representing 4 bytes of all 1s

```
L32 = allOnesPoly[32]
```

$$1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{12} + x^{13} + x^{14} + x^{15} + x^{16} + x^{17} + x^{18} + x^{19} + x^{20} + x^{21} + x^{22} + x^{23} + x^{24} + x^{25} + x^{26} + x^{27} + x^{28} + x^{29} + x^{30} + x^{31}$$

This is the constant pattern the receiver should expect when there are no errors or if the error pattern is undetectable. Basically it is the result of a division by G of the poly representing a sequence of all 1's and shifted left 32 positions.

```
polyCoeffsToHex[PolynomialMod[L32 * x^32, G, Modulus → 2]]
```

$\{1_{16}, c_{16}, 2_{16}, d_{16}, 1_{16}, 9_{16}, e_{16}, d_{16}\}$

the data message as a list of integers (highest order integer is on the left and is transmitted first)

```
expr1 = 256 - i (* the rule for generating data *)

256 - i

listOfIntegers = Table[expr1, {i, 1, 32}] (* generates a list of the values of expr1 when i runs from 1 to 9 *)

{255, 254, 253, 252, 251, 250, 249, 248, 247, 246, 245, 244, 243, 242,
 241, 240, 239, 238, 237, 236, 235, 234, 233, 232, 231, 230, 229, 228, 227, 226, 225, 224}
```

the data message as a polynomial (highest order bit transmitted first) corresponding to the list of integers

```
bitsPerWord = 8 (* each integer will be represented by an 8 bit binary word *)

8
```

```
base = 2 (* in base 2 *)
```

```
2
```

```
F = IntegersToPoly[listOfIntegers, base, bitsPerWord]
```

$$x^{5}+x^{6}+x^{7}+x^{8}+x^{13}+x^{14}+x^{15}+x^{17}+x^{21}+x^{22}+x^{23}+x^{24}+x^{25}+x^{29}+x^{30}+x^{31}+x^{34}+x^{37}+x^{38}+x^{39}+x^{40}+x^{42}+x^{45}+x^{46}+x^{47}+$$
$$x^{49}+x^{50}+x^{53}+x^{54}+x^{55}+x^{56}+x^{57}+x^{58}+x^{61}+x^{62}+x^{63}+x^{67}+x^{69}+x^{70}+x^{71}+x^{72}+x^{75}+x^{77}+x^{78}+x^{79}+x^{81}+x^{83}+x^{85}+x^{86}+$$
$$x^{87}+x^{88}+x^{89}+x^{91}+x^{93}+x^{94}+x^{95}+x^{98}+x^{99}+x^{101}+x^{102}+x^{103}+x^{104}+x^{106}+x^{107}+x^{109}+x^{110}+x^{111}+x^{113}+x^{114}+x^{115}+x^{117}+$$
$$x^{118}+x^{119}+x^{120}+x^{121}+x^{122}+x^{123}+x^{125}+x^{126}+x^{127}+x^{132}+x^{133}+x^{134}+x^{135}+x^{136}+x^{140}+x^{141}+x^{142}+x^{143}+x^{145}+x^{148}+x^{149}+$$
$$x^{150}+x^{151}+x^{152}+x^{153}+x^{156}+x^{157}+x^{158}+x^{159}+x^{162}+x^{164}+x^{165}+x^{166}+x^{167}+x^{168}+x^{170}+x^{172}+x^{173}+x^{174}+x^{175}+x^{177}+x^{178}+$$
$$x^{180}+x^{181}+x^{182}+x^{183}+x^{184}+x^{185}+x^{186}+x^{188}+x^{189}+x^{190}+x^{191}+x^{195}+x^{196}+x^{197}+x^{198}+x^{199}+x^{200}+x^{203}+x^{204}+x^{205}+x^{206}+$$
$$x^{207}+x^{209}+x^{211}+x^{212}+x^{213}+x^{214}+x^{215}+x^{216}+x^{217}+x^{219}+x^{220}+x^{221}+x^{222}+x^{223}+x^{226}+x^{227}+x^{228}+x^{229}+x^{230}+x^{231}+x^{232}+$$
$$x^{234}+x^{235}+x^{236}+x^{237}+x^{238}+x^{239}+x^{241}+x^{242}+x^{243}+x^{244}+x^{245}+x^{246}+x^{247}+x^{248}+x^{249}+x^{250}+x^{251}+x^{252}+x^{253}+x^{254}+x^{255}$$

Here I am testing my new function that is supposed to be the inverse of IntegersToPoly once I get the function right! This function will serve as a double check that my IntegersToPoly is working properly. Ignore this for now.

```
PolyToIntegers[F, 8]
(* I need to specify the number of 8-bit words that the poly represents. My padding appears to be wrong*)
```

```
{0, 0, 0, 0, 255, 254, 253, 252, 251, 250, 249, 248, 247, 246, 245, 244, 243,
 242, 241, 240, 239, 238, 237, 236, 235, 234, 233, 232, 231, 230, 229, 228, 227, 226, 225, 224}
```

At this point I need to reverse bits within each byte to get the poly that is involved in the CRC calculation. I call this process reflection for consistency with the popular CRC literature.

```
ReflectedF = reverseByteBits[F]
```

$$1+x+x^{2}+x^{8}+x^{9}+x^{10}+x^{15}+x^{16}+x^{17}+x^{18}+x^{22}+x^{24}+x^{25}+x^{26}+x^{30}+x^{31}+x^{32}+x^{33}+x^{34}+x^{37}+x^{40}+x^{41}+x^{42}+x^{45}+x^{47}+$$
$$x^{48}+x^{49}+x^{50}+x^{53}+x^{54}+x^{56}+x^{57}+x^{58}+x^{61}+x^{62}+x^{63}+x^{64}+x^{65}+x^{66}+x^{68}+x^{72}+x^{73}+x^{74}+x^{76}+x^{79}+x^{80}+x^{81}+x^{82}+x^{84}+$$
$$x^{86}+x^{88}+x^{89}+x^{90}+x^{92}+x^{94}+x^{95}+x^{96}+x^{97}+x^{98}+x^{100}+x^{101}+x^{104}+x^{105}+x^{106}+x^{108}+x^{109}+x^{111}+x^{112}+x^{113}+x^{114}+x^{116}+$$
$$x^{117}+x^{118}+x^{120}+x^{121}+x^{122}+x^{124}+x^{125}+x^{126}+x^{127}+x^{128}+x^{129}+x^{130}+x^{131}+x^{136}+x^{137}+x^{138}+x^{139}+x^{143}+x^{144}+x^{145}+x^{146}+$$
$$x^{147}+x^{150}+x^{152}+x^{153}+x^{154}+x^{155}+x^{158}+x^{159}+x^{160}+x^{161}+x^{162}+x^{163}+x^{165}+x^{168}+x^{169}+x^{170}+x^{171}+x^{173}+x^{175}+x^{176}+x^{177}+$$
$$x^{178}+x^{179}+x^{181}+x^{182}+x^{184}+x^{185}+x^{186}+x^{187}+x^{189}+x^{190}+x^{191}+x^{192}+x^{193}+x^{194}+x^{195}+x^{196}+x^{200}+x^{201}+x^{202}+x^{203}+x^{204}+$$
$$x^{207}+x^{208}+x^{209}+x^{210}+x^{211}+x^{212}+x^{214}+x^{216}+x^{217}+x^{218}+x^{219}+x^{220}+x^{222}+x^{223}+x^{224}+x^{225}+x^{226}+x^{227}+x^{228}+x^{229}+x^{232}+$$
$$x^{233}+x^{234}+x^{235}+x^{236}+x^{237}+x^{239}+x^{240}+x^{241}+x^{242}+x^{243}+x^{244}+x^{245}+x^{246}+x^{248}+x^{249}+x^{250}+x^{251}+x^{252}+x^{253}+x^{254}+x^{255}$$

Next I calculate the remainder of the division, by G, of the polynomial obtained by adding 32 1's to the most significant positions of the poly ReflectedF after shifting left 32 positions to make room for the frame check sequence. If you can't follow the sentence look at the equation. Addition of 32 1's to the most significant positions of the dividend is equivalent to initializing the CRC register to 1's. Addition is in mod 2 arithmetic which is equivalent to XORing.

```
k = Exponent[ReflectedF, x] + 1
```

```
256
```

```
R = PolynomialMod[ReflectedF * x^32 + L32 * x^k, G, Modulus → 2]  (*this is the remainder of the division*)
```

$$x+x^{3}+x^{7}+x^{11}+x^{12}+x^{14}+x^{15}+x^{19}+x^{20}+x^{22}+x^{23}+x^{25}+x^{27}+x^{28}$$

now I need to reflect the remainder to form the FCS that the iSCSI spec shows

```
ReflectedR = reverseByteBits[R]
```

$$1+x^{4}+x^{6}+x^{8}+x^{9}+x^{11}+x^{12}+x^{16}+x^{17}+x^{19}+x^{20}+x^{27}+x^{28}+x^{30}$$

```
FCS = Expand[L32 + ReflectedR, Modulus → 2] (* note how the FCS is formed from the remainder reflected *)
```

$$x+x^{2}+x^{3}+x^{5}+x^{7}+x^{10}+x^{13}+x^{14}+x^{15}+x^{18}+x^{21}+x^{22}+x^{23}+x^{24}+x^{25}+x^{26}+x^{29}+x^{31}$$

the coefficients of FCS in hex:

```
polyCoeffsToHex[FCS] (* iSCSI spec says a7 e4 e4 ae *)
```

$\{a_{16}, 7_{16}, e_{16}, 4_{16}, e_{16}, 4_{16}, a_{16}, e_{16}\}$

```
M = Expand[x^32 * F + FCS, Modulus → 2]  (* the transmitted message *)
```

$$x + x^{2} + x^{3} + x^{5} + x^{7} + x^{10} + x^{13} + x^{14} + x^{15} + x^{18} + x^{21} + x^{22} + x^{23} + x^{24} + x^{25} + x^{26} + x^{29} + x^{31} + x^{37} + x^{38} + x^{39} + x^{40} + x^{45} + x^{46} + x^{47} +$$
$$x^{49} + x^{53} + x^{54} + x^{55} + x^{56} + x^{57} + x^{61} + x^{62} + x^{63} + x^{66} + x^{69} + x^{70} + x^{71} + x^{72} + x^{74} + x^{77} + x^{78} + x^{79} + x^{81} + x^{82} + x^{85} + x^{86} + x^{87} + x^{88} +$$
$$x^{89} + x^{90} + x^{93} + x^{94} + x^{95} + x^{99} + x^{101} + x^{102} + x^{103} + x^{104} + x^{107} + x^{109} + x^{110} + x^{111} + x^{113} + x^{115} + x^{117} + x^{118} + x^{119} + x^{120} + x^{121} +$$
$$x^{123} + x^{125} + x^{126} + x^{127} + x^{130} + x^{131} + x^{133} + x^{134} + x^{135} + x^{136} + x^{138} + x^{139} + x^{141} + x^{142} + x^{143} + x^{145} + x^{146} + x^{147} + x^{149} + x^{150} +$$
$$x^{151} + x^{152} + x^{153} + x^{154} + x^{155} + x^{157} + x^{158} + x^{159} + x^{164} + x^{165} + x^{166} + x^{167} + x^{168} + x^{172} + x^{173} + x^{174} + x^{175} + x^{177} + x^{180} + x^{181} +$$
$$x^{182} + x^{183} + x^{184} + x^{185} + x^{188} + x^{189} + x^{190} + x^{191} + x^{194} + x^{196} + x^{197} + x^{198} + x^{199} + x^{200} + x^{202} + x^{204} + x^{205} + x^{206} + x^{207} + x^{209} + x^{210} +$$
$$x^{212} + x^{213} + x^{214} + x^{215} + x^{216} + x^{217} + x^{218} + x^{220} + x^{221} + x^{222} + x^{223} + x^{227} + x^{228} + x^{229} + x^{230} + x^{231} + x^{232} + x^{235} + x^{236} + x^{237} + x^{238} +$$
$$x^{239} + x^{241} + x^{243} + x^{244} + x^{245} + x^{246} + x^{247} + x^{248} + x^{249} + x^{251} + x^{252} + x^{253} + x^{254} + x^{255} + x^{258} + x^{259} + x^{260} + x^{261} + x^{262} + x^{263} + x^{264} +$$
$$x^{266} + x^{267} + x^{268} + x^{269} + x^{270} + x^{271} + x^{273} + x^{274} + x^{275} + x^{276} + x^{277} + x^{278} + x^{279} + x^{280} + x^{281} + x^{282} + x^{283} + x^{284} + x^{285} + x^{286} + x^{287}$$

```
Error = reverseByteBits[G + x^5 * G]
 (* a linear combination of G and shifted version of G. This error pattern is undetectable when G is the CRC polynomial *)
```

$$x + x^{2} + x^{7} + x^{8} + 2\,x^{9} + 2\,x^{10} + 2\,x^{12} + x^{13} + x^{14} + x^{15} + 2\,x^{16} + x^{17} + x^{19} + 2\,x^{20} + 2\,x^{21} + x^{23} + x^{24} + x^{25} + 2\,x^{27} + 2\,x^{28} + x^{29} + 2\,x^{30} + x^{31} + x^{34} + x^{38} + 2\,x^{39}$$

```
M* = Expand[M + Error, Modulus → 2] (* the sum of the message and the error expressed as a polynomial *)
```

$$x^{3} + x^{5} + x^{8} + x^{10} + x^{17} + x^{18} + x^{19} + x^{21} + x^{22} + x^{26} + x^{34} + x^{37} + x^{39} + x^{40} + x^{45} + x^{46} + x^{47} + x^{49} + x^{53} + x^{54} + x^{55} + x^{56} + x^{57} + x^{61} + x^{62} + x^{63} + x^{66} + x^{69} +$$
$$x^{70} + x^{71} + x^{72} + x^{74} + x^{77} + x^{78} + x^{79} + x^{81} + x^{82} + x^{85} + x^{86} + x^{87} + x^{88} + x^{89} + x^{90} + x^{93} + x^{94} + x^{95} + x^{99} + x^{101} + x^{102} + x^{103} + x^{104} + x^{107} + x^{109} +$$
$$x^{110} + x^{111} + x^{113} + x^{115} + x^{117} + x^{118} + x^{119} + x^{120} + x^{121} + x^{123} + x^{125} + x^{126} + x^{127} + x^{130} + x^{131} + x^{133} + x^{134} + x^{135} + x^{136} + x^{138} + x^{139} + x^{141} +$$
$$x^{142} + x^{143} + x^{145} + x^{146} + x^{147} + x^{149} + x^{150} + x^{151} + x^{152} + x^{153} + x^{154} + x^{155} + x^{157} + x^{158} + x^{159} + x^{164} + x^{165} + x^{166} + x^{167} + x^{168} + x^{172} + x^{173} +$$
$$x^{174} + x^{175} + x^{177} + x^{180} + x^{181} + x^{182} + x^{183} + x^{184} + x^{185} + x^{188} + x^{189} + x^{190} + x^{191} + x^{194} + x^{196} + x^{197} + x^{198} + x^{199} + x^{200} + x^{202} + x^{204} + x^{205} +$$
$$x^{206} + x^{207} + x^{209} + x^{210} + x^{212} + x^{213} + x^{214} + x^{215} + x^{216} + x^{217} + x^{218} + x^{220} + x^{221} + x^{222} + x^{223} + x^{227} + x^{228} + x^{229} + x^{230} + x^{231} + x^{232} + x^{235} +$$
$$x^{236} + x^{237} + x^{238} + x^{239} + x^{241} + x^{243} + x^{244} + x^{245} + x^{246} + x^{247} + x^{248} + x^{249} + x^{251} + x^{252} + x^{253} + x^{254} + x^{255} + x^{258} + x^{259} + x^{260} + x^{261} + x^{262} +$$
$$x^{263} + x^{264} + x^{266} + x^{267} + x^{268} + x^{269} + x^{270} + x^{271} + x^{273} + x^{274} + x^{275} + x^{276} + x^{277} + x^{278} + x^{279} + x^{280} + x^{281} + x^{282} + x^{283} + x^{284} + x^{285} + x^{286} + x^{287}$$

```
polyCoeffsToHex[M*] (* the sum of the message and the error expressed as hex coefficients of a polynomial *)
```

$\{0_{16}, 4_{16}, 6_{16}, e_{16}, 0_{16}, 5_{16}, 2_{16}, 8_{16}\}$

ReflectedM* = reverseByteBits[M*] (* another reflection *)

$$x^{2}+x^{4}+x^{13}+x^{15}+x^{17}+x^{18}+x^{20}+x^{21}+x^{22}+x^{29}+x^{32}+x^{34}+x^{37}+x^{40}+x^{41}+x^{42}+x^{47}+x^{48}+x^{49}+x^{50}+x^{54}+x^{56}+x^{57}+x^{58}+x^{62}+x^{63}+x^{64}+x^{65}+$$
$$x^{66}+x^{69}+x^{72}+x^{73}+x^{74}+x^{77}+x^{79}+x^{80}+x^{81}+x^{82}+x^{85}+x^{86}+x^{88}+x^{89}+x^{90}+x^{93}+x^{94}+x^{95}+x^{96}+x^{97}+x^{98}+x^{100}+x^{104}+x^{105}+x^{106}+$$
$$x^{108}+x^{111}+x^{112}+x^{113}+x^{114}+x^{116}+x^{118}+x^{120}+x^{121}+x^{122}+x^{124}+x^{126}+x^{127}+x^{128}+x^{129}+x^{130}+x^{132}+x^{133}+x^{136}+x^{137}+x^{138}+x^{140}+$$
$$x^{141}+x^{143}+x^{144}+x^{145}+x^{146}+x^{148}+x^{149}+x^{150}+x^{152}+x^{153}+x^{154}+x^{156}+x^{157}+x^{158}+x^{159}+x^{160}+x^{161}+x^{162}+x^{163}+x^{168}+x^{169}+x^{170}+$$
$$x^{171}+x^{175}+x^{176}+x^{177}+x^{178}+x^{179}+x^{182}+x^{184}+x^{185}+x^{186}+x^{187}+x^{190}+x^{191}+x^{192}+x^{193}+x^{194}+x^{195}+x^{197}+x^{200}+x^{201}+x^{202}+x^{203}+$$
$$x^{205}+x^{207}+x^{208}+x^{209}+x^{210}+x^{211}+x^{213}+x^{214}+x^{216}+x^{217}+x^{218}+x^{219}+x^{221}+x^{222}+x^{223}+x^{224}+x^{225}+x^{226}+x^{227}+x^{228}+x^{232}+x^{233}+$$
$$x^{234}+x^{235}+x^{236}+x^{239}+x^{240}+x^{241}+x^{242}+x^{243}+x^{244}+x^{246}+x^{248}+x^{249}+x^{250}+x^{251}+x^{252}+x^{254}+x^{255}+x^{256}+x^{257}+x^{258}+x^{259}+x^{260}+$$
$$x^{261}+x^{264}+x^{265}+x^{266}+x^{267}+x^{268}+x^{269}+x^{271}+x^{272}+x^{273}+x^{274}+x^{275}+x^{276}+x^{277}+x^{278}+x^{280}+x^{281}+x^{282}+x^{283}+x^{284}+x^{285}+x^{286}+x^{287}$$

What follows is the CRC computed at the receiver expressed as a polynomial. Basically I take the received message after reflection and add 32 1's to the most significant bits (this is the effect of initializing the CRC to 1's); then shift left 32 positions and perform the division by G.

Rec = PolynomialMod[x^32 * (ReflectedM* + x^k * L32), G, Modulus → 2] (* the CRC computed at the receiver as a polynomial *)

$$1+x^{2}+x^{3}+x^{5}+x^{6}+x^{7}+x^{8}+x^{11}+x^{12}+x^{16}+x^{18}+x^{19}+x^{21}+x^{26}+x^{27}+x^{28}$$

polyCoeffsToHex[Rec] (* the CRC computed at the receiver as coefficients of a polynomial *)

$\{1_{16}, c_{16}, 2_{16}, d_{16}, 1_{16}, 9_{16}, e_{16}, d_{16}\}$