Experiment 2: Error Detection using CRC-CCITT

Aim: To apply CRC (CCITT Polynomial) for error detection

Objective: After carrying out this experiment, students will be able to:

- Apply CRC CCITT to develop codes for error detection
- Analyse how this CRC is able to detect bit errors irrespective of their length and position in the data

Problem statement: You are required to write a program that uses CRC to detect burst errors in transmitted data. Initially, write the program using the CRC example you studied in class. Your final program should ask the user to input data and choose a generator polynomial from the list given in the figure below. Your program is required to calculate the checksum and the transmitted data. Subsequently, the user enters the received data. Applying the same generator polynomial on the received data should result in a remainder of 0.

Name	Polynomial	Application
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

Analysis: While analyzing your program, you are required to address the following points:

- How is this method different from 2D parity scheme that you have implemented previously?
- What are the limitations of this method of error detection?

MARKS DISTRIBUTION

Component	Maximum Marks	Marks Obtained
Preparation of Document	7	
Results	7	
Viva	6	
Total	**20**	

Submitted by:

Register No:

1. Algorithm/Flowchart

getCrc(char *input,char *key,char sr)

step1:- start

step2:-declare int I,j,keylen,msglen and char temp[30], quot[100], rem[30], key1[30];

step3:- keylen <-strlen(key)
msglen <- strlen(input)
strcpy(key1, key)

// division with help of xor
Step4:- for i=0 to keylen-1, do
4.1) input[msglen + i] <- '0'

Step5:- for i = 0 to keylen, do
5.1) temp[i] <- input[i]
Step6:-
for i = 0 to msglen, do
6.1) quot[i] <- temp[0]
6.2) if (quot[i] == '0') do:
for j = 0 to keylen,do
key[j] <- '0'; else do:
for j = 0 to keylen, do
key[j] <- key1[j]

6.3) for j = keylen – 1 till j > 0, do

if (temp[j] == key[j])
rem[j - 1] <- '0'
else
rem[j - 1] <- '1'

6.4) rem[keylen - 1] <-input[i + keylen]
6.5) strcpy(temp, rem)

Step 7:- strcpy(rem, temp)

```
Step 8:- write ("\n Remainder is ")
Step 9:- for i = 0 to  keylen - 1
                write ("%c", rem[i]);

// check if reciver or sender
// if sender then print the data
// if reciever than check if remainder is 0 or not

Step10:-
 if (sr=='s') then do
    10.1) write ("\n Transmitted data is: ")
   10.2) for i = 0 to msglen
                write ("%c", input[i])
    10.3)for i = 0 to keylen - 1
                write ("%c", rem[i])
Step11:-
 if (sr=='r') then do:
    11.1)for i = 0 to keylen - 1
              11.1.1)if(rem[i]!='0') then do:
                            break
    11.2)       (i==(keylen-1))
                      ? write ("\n Since Remainder is 0\n Recieved data is  correct")
                      : write("\n Since Remainder is not 0\n Recieved data is not correct ");

Step12:- stop

 Void main()
Step1: start
Step2:-Declare int dataSize,gSize,ch
        Declare char data[100],generator[100],data1[100]
Step3:- write("Select the GP \n1)CRC-8 \n 2)CRC-10 \n 3)CRC-16 \n 4)CRC-32 \n")
Step4:- write ("Enter Choice:- ")
Step5:- read(" %d",&ch)
Step6:- switch (ch)
           case 1:
              strcpy(generator,"101011000")
              gSize=9;
              then break
           case 2:
              strcpy(generator,"10110101001")
```


gSize=11;
then break
case 3:
strcpy(generator,"10010110101001101")
gSize=17
then break
case 4:
strcpy(generator,"100101101010011011001011010100110")
gSize=33
then break
default:
write ("Enter Valid Input")
then break

Step7:- write ("\n The GP is %s ",generator)
Step8:- write ("\n The size of the GP is %d \n",gSize)

// sender side
Step9:-
Call getCrc(data,generator,'s')

//reciever side
Step10:- write ("\n Enter Recieved Data:-")
Step11:- char data2[100]
Step12:- scanf(" %s",data2)
Step13:- Call getCrc(data2,generator,'r')

2. Program

CRC :-

```
#include <stdio.h>
#include <string.h>
void getCrc(char *input,char *key,char sr);
void main() {
    int dataSize,gSize,ch;
    char data[100],generator[100],data1[100];

    printf("Enter binary bits:- ");
    gets(data);

    printf("Select the GP \n1)CRC-8 \n 2)CRC-10 \n 3)CRC-16 \n 4)CRC-
32 \n");
```


```
    printf("Enter Choice:- ");
    scanf(" %d",&ch);
    switch (ch)
    {
    case 1:
        strcpy(generator,"101011000");
        gSize=9;
        /* code */
        break;
    case 2:
        strcpy(generator,"10110101001");
        gSize=11;
        break;
    case 3:
        strcpy(generator,"10010110101001101");
        gSize=17;
        break;
    case 4:
        strcpy(generator,"10010110101001101100101101010100110");
        gSize=33;
        break;
    default:
        printf("Enter Valid Input");
        break;
    }
    printf("\n The GP is %s ",generator);
    printf("\n The size of the GP is %d \n",gSize);

    // sender side
    getCrc(data,generator,'s');

    //reciever side
    printf("\n Enter Recieved Data:-");
    char data2[100];
    scanf(" %s",data2);

    getCrc(data2,generator,'r');

    // return 0;
}

void getCrc(char *input,char *key,char sr){
    int i, j, keylen, msglen;
    char temp[30], quot[100], rem[30], key1[30];
```


```
    keylen = strlen(key);
    msglen = strlen(input);
    strcpy(key1, key);
    // division
    for (i = 0; i < keylen - 1; i++)
    {
        input[msglen + i] = '0';
    }
    for (i = 0; i < keylen; i++)
        temp[i] = input[i];
    for (i = 0; i < msglen; i++)
    {
        quot[i] = temp[0];
        if (quot[i] == '0')
            for (j = 0; j < keylen; j++)
                key[j] = '0';
        else
            for (j = 0; j < keylen; j++)
                key[j] = key1[j];
        for (j = keylen - 1; j > 0; j--)
        {
            if (temp[j] == key[j])
                rem[j - 1] = '0';
            else
                rem[j - 1] = '1';
        }
        rem[keylen - 1] = input[i + keylen];
        strcpy(temp, rem);
    }
    strcpy(rem, temp);
    printf("\n Remainder is ");
    for (i = 0; i < keylen - 1; i++)
        printf("%c", rem[i]);

    // check if reciver or sender
    // if sender then print the data
    // if reciever than check if remainder is 0 or not
    if (sr=='s')
    {
        printf("\n Transmitted data is: ");
        for (i = 0; i < msglen; i++)
            printf("%c", input[i]);
        for (i = 0; i < keylen - 1; i++)
            printf("%c", rem[i]);
```


```
    }
    if (sr=='r')
    {
        for (i = 0; i < keylen - 1; i++){
            if(rem[i]!='0'){
                break;
            }
        }
        (i==(keylen-1))
        ?printf("\n Since Remainder is 0\n Recieved data is  correct")
        :printf("\n Since Remainder is not 0\n Recieved data is not correc
t ");
    }
}
```

3. Results

CRC output:-

Received without error:-

```
Enter binary bits:- 10110111011000
Select the GP
1)CRC-8
 2)CRC-10
 3)CRC-16
 4)CRC-32
Enter Choice:- 1

 The GP is 101011000
 The size of the GP is 9

 Remainder is 10001000
 Transmitted data is: 10110111011000100010001000
 Enter Recieved Data:-10110111011000100010001000

 Remainder is 00000000
 Since Remainder is 0
 Recieved data is  correct
```


Received with error:-

```
Enter binary bits:- 10110111011000
Select the GP
1)CRC-8
 2)CRC-10
 3)CRC-16
 4)CRC-32
Enter Choice:- 1

 The GP is 101011000
 The size of the GP is 9

 Remainder is 10001000
 Transmitted data is: 1011011101100010001000
 Enter Recieved Data:-1011000000000010001000

 Remainder is 01011000
 Since Remainder is not 0
 Recieved data is not correct
```

4. Analysis and Discussions

Intro to CRC:-

CRC or Cyclic Redundancy Check is a method of detecting accidental changes/errors in the communication channel.

CRC uses Generator Polynomial which is available on both sender and receiver side. An example generator polynomial is of the form like x3 + x + 1. This generator polynomial represents key 1011. Another example is x2 + 1 that represents key 101.

n : Number of bits in data to be sent

from sender side.

k : Number of bits in the key obtained

from generator polynomial.

Sender Side (Generation of Encoded Data from Data and Generator Polynomial (or Key)):

The binary data is first augmented by adding k-1 zeros in the end of the data

Use modulo-2 binary division to divide binary data by the key and store remainder of division.

Append the remainder at the end of the data to form the encoded data and send the same

.

Receiver Side (Check if there are errors introduced in transmission)
Perform modulo-2 division again and if the remainder is 0, then there are no errors

What is the difference between parity and cyclic redundancy check?

- Adding a parity bit is just one special case of performing a cyclic redundancy check (CRC).
- "The simplest error-detection system, the parity bit, is in fact a 1-bit CRC: it uses the generator polynomial x + 1 (two terms), and has the name CRC-1."

CRC over parity:-

- For instance, it can detect all single bit errors, all double bit errors, any odd number of errors, and most burst errors. Parity check, on the other hand, can only detect single bit errors, while checksum can detect all single bit and some multiple bit errors. Obviously, CRC is the most robust of the group.

5. Conclusions

   By considering above points, it can be said that CRC is more reliable than parity error detection method.

6. Comments

   a. Limitations of the experiment
      - Though the Cyclic Redundancy Check look like an authentication mechanism, it is non-trivial and easy to crack mechanism. It is not suitable for security purpose.
      - Without the error correcting mechanism using CRC alone will be a useless thing.

   b. Learning:-
      Learned various aspects of Cyclic Redundancy Check is a method of detecting accidental changes/errors.