Com S 352 Fall 2017

Project 2

Secure Communication between Client and Multi-Processing Server using Socket Programming

100 points

Due: Friday, November 17, 11:59pm

Socket Programming

Problem Description

The purpose of this programming assignment is to practice with Socket Programming. In this project, you would be required to write "C" programs for "Client" and "Server" using socket programming to implement secure communication between clients and a multi-processing server. On Server side, use multi-processing, such that Server can handle requests from more than one client, using one process to handle each client's request. A multi-processing server creates a new process for each communication it accepts from a client.

In this programming assignment, you will be asked to implement a simple client/server program, using connection-oriented communication (**socket programming using TCP**).

1. Client:

- Input a string (e.g. system time, IP address) called "A's data package" hereafter
- Client calculates the hash value as:

  hash(A's data package) = message digest
- Create "Digital Signature" (by encrypting the message digest using the StringEncoder.c program provided at the end of this document) as:

  Encrypt (message digest) = digital signature
- Client sends the original string along with its digital signature to the server.

2. Server:

- Receive the string and its digital signature from the client,
- Server creates a “New Process” to handle the client,
- Server calculates the hash value over the string as:

  hash(A’s data package) = message digest’
- Server creates digital signature’ (by encrypting message digest’ using the StringEncoder.c program provided at the end of this document):

  Encrypt(message digest’) = digital signature’
- Return "true" to the client if the received “digital signature” match with the signature generated at server i.e. digital signature’, otherwise return "false" to the client.

  return compare(digital signature, digital signature’)

3. Client displays the result (“true” or ‘’false”) received from the Server on its screen.

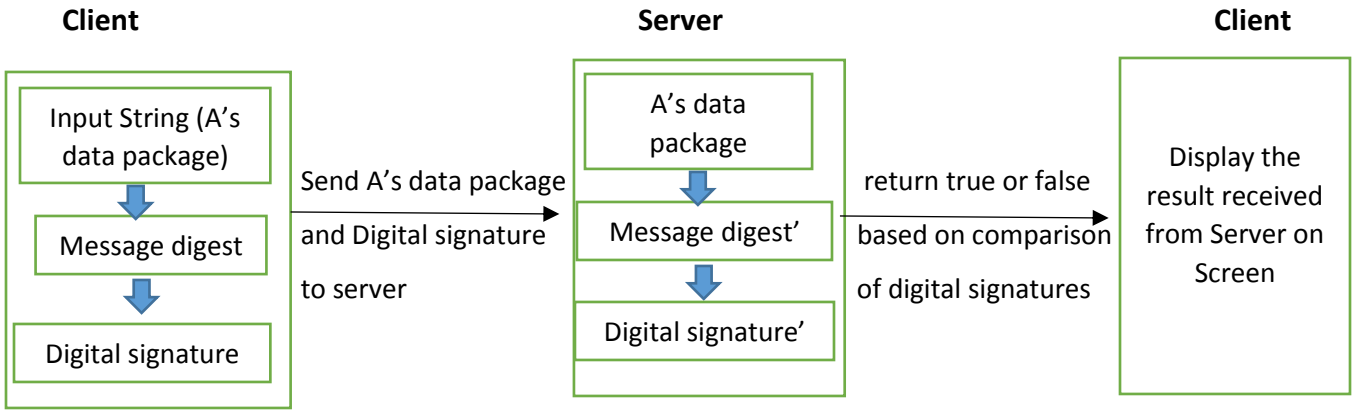

Figure 1:

Hints:

- To calculate the Hash, use Secure Hash Algorithm (SHA-1). One way to do this in C is to use the OpenSSL library.
- The encryption at the client and server can be done using the StringEncoder.c program (provided at the end of this document).

Extra Credit: ***20 points***

Multi-Threaded Server (instead of Multi-Processing Server)

Implement secure communication between clients and a multi-threaded server. On Server side, use multi-threading, such that Server can handle requests from more than one client, using one thread to handle each client's request. A multi-threaded server creates a new thread for each communication it accepts from a client. A thread is a sequence of instructions that run independently of the program and of any other threads.

Note: The extra credit requires the same client-server communication implemented using threads instead of processes on the server side (all other details/specifications remain the same).

Submission

- You should use C to develop the code.
- You need to turn in electronically by submitting a zip file named Firstname_Lastname_Project2.zip.
- Source code must include proper documentation to receive full credit.
- All projects require the use of a make file or a certain script file (accompanying with a readme file to specify how to use the script file to compile), such that the grader will be able to compile/build your executable by simply typing "make" or some simple command that you specify in your readme file.
- Source code must compile and run correctly on the department machine "pyrite", which will be used by the TA for grading.
- You are responsible for thoroughly testing and debugging your code. The TA may try to break your code by subjecting it to bizarre test cases.
- You can have multiple submissions, but the TA will grade only the last one.

Start as early as possible!

StringEncoder.c

```
#include <stdio.h>
#include <string.h>
#include <math.h>

//This map structure will be used to store the encodings
//Here, we have used decimal representation of the corresponding 3-bit
binary numbers.

int binaryMap[8] = {6,7,5,4,3,2,0,1};


//inplace_reverse takes string as an input and reverses the contents
of the string inplace.
void inplace_reverse(char * str)
{
  if (str)
  {
    char * end = str + strlen(str) - 1;

    // swap the values in the two given variables
    // XXX: fails when a and b refer to same memory location
#   define XOR_SWAP(a,b) do\
    {\
      a ^= b;\
      b ^= a;\
      a ^= b;\
    } while (0)

    // walk inwards from both ends of the string,
    // swapping until we get to the middle
    while (str < end)
    {
      XOR_SWAP(*str, *end);
      str++;
      end--;
    }
#   undef XOR_SWAP
  }
}


//binaryToDecimal will return the decimal number for the corresponding
binary form. for eg. decimal [5] for binary [101].
int binaryToDecimal(long n)
{
    int decimalNumber = 0, i = 0, remainder;
    while (n!=0)
    {
        remainder = n%10;
```

```
        n /= 10;
        decimalNumber += remainder*pow(2,i);
        ++i;
    }
    return decimalNumber;
}


//decimalToBinary will return the binary form for the corresponding
decimal form. for eg. binary [101] for decimal [5]
long decimalToBinary(long n) {
 int remainder;
 long binary = 0, i = 1;

    while(n != 0) {
        remainder = n%2;
        n = n/2;
        binary= binary + (remainder*i);
        i = i*10;
    }
    return binary;
}


//stringToAscii will print ASCII representation of a given string
void stringToAscii(char *s){
  char *ptr = s;
  char finalRes[256]="";

  //Go through each character in the string and append the
correpsonding ASCII representaion to the final result
  while(*ptr){
    int asciiDecimal = (int)*ptr;
    int res = decimalToBinary(asciiDecimal);

    char snum[5];
    sprintf(snum, "%d",res);
    strcat(finalRes,snum);

    ptr++;
  }

  //print the complete ASCII form
  printf("\nstringToAscii:: %s",finalRes);
  return;
}




//stringToReverseAscii will print the reverse ASCII representation of
the given string
```

```
void stringToReverseAscii(char *s){
  char *ptr = s;
  char finalRes[256]="";

  //Visit each character in the string and append its ASCII
representation to a temp result
  while(*ptr){
    int asciiDecimal = (int)*ptr;
    int res = decimalToBinary(asciiDecimal);

    char snum[5];
    sprintf(snum, "%d",res);
    strcat(finalRes,snum);

    ptr++;
  }

  //Reverse the temp result to get the final answer
  inplace_reverse(finalRes);

  printf("\nstringToReverseAscii:: %s",finalRes);
  return;
}


//stringToEncodedAscii prints the encoded representaion of the ASCII
representaion of a given string(Encoding is based on encoding array at
the top of this file)
void stringToEncodedAscii(char *s){
  char *ptr = s;
  char finalRes[256]="";

  //First, get the ASCII representation of the given string
  while(*ptr){
    int asciiDecimal = (int)*ptr;
    int res = decimalToBinary(asciiDecimal);

    char snum[5];
    sprintf(snum, "%d",res);
    strcat(finalRes,snum);

    ptr++;
  }

  //Since, we are encoding 3 bits at a time of the ASCII
representaion, we need to pad extra 0s to the ASCII string to make
  //its length a multiple of 3.
  int lenOfFinalRes = strlen(finalRes);
  if((lenOfFinalRes % 3) != 0){
    while((lenOfFinalRes % 3) != 0){
      strcat(finalRes,"0");
      lenOfFinalRes++;
```

```
    }
  }

  //printf("\nString length : %d\n",lenOfFinalRes);


  char *newPtr = finalRes;
  int j,k;
  char finalResEncoded[256] = "";

  //We now take 3 bits at a time, find the encoding from encoding
table at top and append the encoding to a new result string.
  for(j=0;j<(lenOfFinalRes/3);j++){
     char blk[10]= "";
     for(k=0;k<3;k++){
       //printf("\n next : %c\n",*newPtr);
       char ch[2];
       sprintf(ch, "%c",*newPtr);
       strcat(blk,ch);
       newPtr++;
     }

    long ret;
    char *ptr1;
    ret = strtol(blk, &ptr1, 10);
    int retI = binaryToDecimal(ret);
    int encodedRet = binaryMap[retI];
    int res = decimalToBinary(encodedRet);

    char snum[5];
    sprintf(snum, "%d",res);

     //The following is to make sure that, we always have a 3 bit
binary representaion. For eg. decimal 3 is representated as 011 and
not 11
    char paddedStr[5] = "";
    sprintf(paddedStr, "%s","");
    int snumLen = strlen(snum);
    if((snumLen%3) != 0){
      int z;
      for(z=0; z < (3-(snumLen%3)); z++){
        strcat(paddedStr,"0");
      }
    }

    strcat(paddedStr,snum);

   // printf("\nnormal : %ld ; retI : %d; encodedRet : %d; encoded :
%s",ret,retI,encodedRet ,paddedStr);
    strcat(finalResEncoded,paddedStr);
  }
```

```
  printf("\nstringToEncodedAscii :: %s",finalResEncoded);

}

int main() {

  char s[256];
  printf("enter the string : ");
  scanf("%s", s);

  stringToAscii(s);
  stringToReverseAscii(s);
  stringToEncodedAscii(s);
  return 0;
}
```