wm054@cs2:~$ cat DHListen.cpp

// tcplisten.cpp

//

// a TCP time server

//

// This program is a TCP server servicing port 1400 that returns, in readable

// form, the time on this machine. All that a client has to do is to connect

// and send the word 'time', then read what is sent back.

//

#include <iostream>

#include <string>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <fcntl.h>

#include <stdlib.h>

#include <math.h>

#include <sstream>

#include <errno.h>

#include <string.h>

#include <sys/wait.h>

#include <time.h>

#define SIZE 1024

#define TIME\_PORT 20054

using namespace std;

char buf[SIZE];

int main(int argc, char \*argv[])

{

int p = 7;

int g = 6; //0<g<p

int x1 = rand() % (p-1); //0<x1<p-1

int y1 = pow(g, x1); //g^x1

y1 = y1 % p; // %p

char \*ToSend;

int size;

cout << "p = " << p << endl;

cout << "g = " << g << endl;

cout << "x1 = " << x1 << endl;

cout << "y1 = " << y1 << endl;

unsigned int len;

int sockfd, client\_sockfd, buflen;

int nread;

struct sockaddr\_in serv\_addr, client\_addr;

time\_t t;

struct tm \*stmp;

if ((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

{ cout << "error with socket()"; exit(2); }

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_addr.s\_addr = INADDR\_ANY;

serv\_addr.sin\_port = htons(TIME\_PORT);

bzero(&(serv\_addr.sin\_zero),8);

t = bind(sockfd, (struct sockaddr \*)&serv\_addr, sizeof(struct sockaddr\_in)) ;

if (t == -1)

{ cout << "error with Bind()" << endl;

exit(0);

}

cout << "tcplisten::bind was successful\n" << endl;

len = sizeof(struct sockaddr\_in);

listen(sockfd, 5);

cout << "tcplisten::listen() was successful" << endl;

cout << "LINE " << \_\_LINE\_\_ << endl;

cout << "sockfd " << sockfd << endl;

client\_sockfd = accept(sockfd, (sockaddr \*)&client\_addr, &len);

cout << "LINE " << \_\_LINE\_\_ << endl;

cout << "tcplisten::connection made - client sockfd is "

<< client\_sockfd << endl;

buflen = read(client\_sockfd,&buf,SIZE);

cout << "tcplisten::Talker-y2: " << buf << endl;

//write(1,buf, buflen);

cout << endl;

ostringstream OSS;

string AString = "";

OSS << y1;

AString = OSS.str();

size = AString.length();

ToSend = new char[size + 1];

strcpy(ToSend, AString.c\_str());

write(client\_sockfd, ToSend, size);

int y2 = atoi(buf);

int z1 = pow(y2, x1);

z1 = z1 % p;

//len = sizeof(struct sockaddr\_in);

//listen(sockfd, 5);

//cout << "tcplisten::listen() was successful" << endl;

//client\_sockfd = accept(sockfd, (sockaddr \*)&client\_addr, &len);

//cout << "tcplisten::connection made - client sockfd is "

// << client\_sockfd << endl;

memset(buf, 0, SIZE);

buflen = read(client\_sockfd,&buf,SIZE);

buf[buflen] = 0;

cout << "Encrypted Message: " << buf << endl;

//write(1,buf, buflen);

cout << endl;

string ToDec=buf;

size = ToDec.length();

char\* Decrypted = new char[size+1];

for(int x = 0; x < size; x++)

{

Decrypted[x] = ToDec[x] ^ z1;

}

cout << "Decrypted Message: " << Decrypted << endl;

write(client\_sockfd, "Got It", 6);

close(client\_sockfd);

close(sockfd);

}