International Institute of Information Technology Hyderabad

System and Network Security (CS5470)

Lab Assignment 1:
Secure file transfer
using a client-server programming model
with RSA public key cryptosystem along with SHA-1 hash function
Deadline: January 30, 2016 (Saturday), 23:59 PM
Total Marks: 100

Note:- *It is strongly recommended that no student is allowed to copy programs from others. Hence, if there is any duplicate in the assignment, simply both the parties will be given zero marks without any compromisation. Rest of assignments will not be evaluated further and assignment marks will not be considered towards final grading in the course. No assignment will be taken after deadline. Name your programs as rollno_assign_1_client.c and rollno_assign_1_server.c for the client and server, respectively. Upload your only rollno_assign_1_client.c and rollno_assign_1_server.c files along with a README file in a zip file (rollno_assign_1.zip) to course portal (moodle).* ***You are restricted to use only C programming language for implementation. No other programming language implementation will be accepted.***

Problem Description

1. Suppose a user A (client) wants to securely receive a file stored at the database of the user B (server).

2. The client A first generates a pair of public and private keys, say $\{(e, n), (d, n)\}$ using the RSA key generation algorithm. After that A sends a message PUBKEY consisting of the public key of A to the server B. Thus, the client A keeps his/her own private key (d, n).

3. The client A sends a request message, say REQ for a file to be sent securely by the server B. If the file is not present at the server B, an appropriate message (DISCONNECT) to be sent by the server B indicating that *no such file exists*.

4. Otherwise, assume that the transmitted file is formed from a character set containing the following characters only. Encoding of these characters are given in Table 1.

   Use the RSA algorithm with the one-way hash function SHA-1 as follows. The server B encodes the characters in the file using the encoding technique provided in Table 1. Then, based on the value of n, which is $n = p \times q$, B prepares all the blocks, say n blocks for the file. Each block, say P_i

Table 1: Encoding rule

Character	Value	Character	Value
blank space	00	A	01
B	02	C	03
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Y	24	Z	25
a	26	b	27
$\vdots$	$\vdots$	$\vdots$	$\vdots$
y	50	z	51
0	52	1	53
$\vdots$	$\vdots$	$\vdots$	$\vdots$
8	59	9	60
,	61	.	62
!	63		

$(i = 1, 2, \ldots, n)$ is encrypted using the RSA encryption algorithm $E(\cdot)$ using the public key (e, n) of A to produce the corresponding ciphertext block, say C_i as $C_i = E_e(P_i) = P_i^e \pmod{n}$, and the corresponding the hash digest $HD_{P_i} = h(P_i)$, where $h(\cdot)$ is SHA-1. B then sends a message REP consisting of C_i and HD_{P_i}. This process is repeated until the entire file is securely transferred to A and at the end, the server B sends REQCOM message (request complete) to the client A.

Note that for any REP message, the client A first decrypts C_i using his/her private key (d, n) as $P_i' = D_d(C_i) = C_i^d \pmod{n}$ and then recomputes $HD'_{P_i} = h(P_i')$. If there is a mis-match between HD_{P_i} and HD'_{P_i}, the client A aborts the session by sending an appropriate message (DISCONNECT) to the server B. Otherwise, the client A stores the recovered original plaintext (after decoding) to the same file as requested in the REQ message. In this way, the client A will have the entire file from the server B.

5. Finally, a disconnect message (DISCONNECT) is exchanged between the client A and the server B, if no more file needs to be transmitted from the server B.

The protocol messages to be used in implementation are provided in Table 2.

Table 2: Protocol messages to be used in implementation

Opcode	*Message*	*Description*
10	PUBKEY	Public key (e, n) sent to the server by the client
20	REQ	Request for service (transferring a requested file) from the server to the client
30	REP	Sending (C_i, HD_{P_i}) for each plaintext block P_i from the server to the client
40	REQCOM	Request completion message from the server to the client
50	DISCONNET	Disconnect request message sent from the client to the server

The communication message flow for this problem are given in Table 3.

Table 3: Communication message flow to be used for implementation

Client (User A)	Server (User B)
RSA key generation	
$\xrightarrow{PUBKEY}$	
Secure file transfer using RSA and SHA-1:	
$\xrightarrow{REQ}$	
	$\xleftarrow{REP}$
	$\xleftarrow{REP}$
	$\vdots$
	$\xleftarrow{REQCOM}$
$\xrightarrow{DISCONNET}$	
	$\xleftarrow{DISCONNET}$

Data structure to be used for implementation

```
/* Header of a general message */
typedef struct {
    int opcode; /* opcode for a message */
    int s_addr; /* source address */
    int d_addr; /* destination address */
} Hdr;
```

```
/* RSA public key */
typedef struct {
    long e; /* encryption exponent */
    long n; /* modulus */
} PubKey;
```

Similarly, define other message structures clearly. Finally, the generalized message will contain the Hdr and union of all the other messages defined. For example,

```
/*A general message */
typedef struct {
    Hdr hdr; /* Header for a message */
    typedef union {
    PubKey pubkey;
    Req req;
    Rep rep;
    Disconnect disconnect;
    Reqcom reqcom;
    } AllMsg;
} Msg;
```

Instructions:

1. Write your server.c program in server directory. Run the program as: ./a.out
2. Write your client.c program in client directory. Run the program as: ./a.out 127.0.0.1 (for local host loop back)
3. Display your all messages at both client and server sides.
4. Every student must register in course portal (moodle) for online submissions.