International Institute of Information Technology Hyderabad

Systems and Network Security (CS5470)

Lab Assignment 2: Implementation of the Digital Signature Algorithm (DSA) using a client-server program

Deadline: January 28, 2011 (Friday)

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 roll_no_assign_2_client.c and roll_no_assign_2_server.c for the client and server. Upload your only client.c and server.c files in a zip/tar file to course portal. If you are using other programming languages other than C, they are also welcomed.*

Description of Problem

See the Digital Signature Algorithm for details.

1. The client (Alice) generates a large prime p using the Miller Robin Primality Test algorithm. Alice then selects another prime q such that $q|(p-1)$. Alice computes $g = h^{(p-1)/q} \mod p$ for $1 < h < (p-1)$ such that $g > 1$.

2. Alice selects randomly a private key x such that $0 < x < q$.

3. Alice computes the public key $y = g^x \mod p$.
   Alice sends $PUBKEY(ID_{Alice}, p, q, g, y, h(\cdot))$ to server (Bob).

4. Alice generates randomly a secret number k per message such that $0 < k < q$. Alice then computes the signature on a message m as follows:

   4.1. Computes $r = (g^k \mod p) \mod q$.
   4.2. Computes $s = [k^{-1}(H(m) + x.r)] \mod q$.
   4.3. The signature is (r, s).

   Alice sends the signed message $SIGNEDMSG(ID_{Alice}, m, signature)$ to Bob.

5. After receiving the signed message from Alice, Bob verifies the signature using the signature verification algorithm given in the Digital Signature Algorithm (DSA). If signature is valid set status as true; otherwise set status as false.
   Bob sends the message $VERSTATUS(ID_{Bob}, status)$ with signature verification result to Alice.

Protocol Messages to be used in implementation

Opcode	*Message*	*Description*
10	PUBKEY	public key along with global elements sent to the server by the client
20	SIGNEDMSG	The message along with signature sent from client to server
30	VERSTATUS	Signature verification status by the server

Data structure to be used in implementation

```
#define MAX_SIZE 20
#define MAX_LEN 1024

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

/* Signature of a message m */
typedef struct {
    int r;
    int s;
} Signature;

/*A general message */
typedef struct {
    Hdr hdr; /* Header for a message */
    char ID[MAX_SIZE]; /* The identifier of a user */
    int p; /* A large prime */
    int q; /* A prime factor of (p-1)*/
    int g; /* g = h^{(p-1)/q} mod p, with 1 < h < (p-1) and g > 1 */
    int y; /* A public key generated by user */
    char hash_algo[MAX_SIZE];  Type of hash algorithm is used */
    char plaintext[MAX_LEN]; /* Contains a plaintext message*/
    Signature sign; /* Contains the signature on a plaintext message*/
    int ver_status; /* Successful or unsuccessful result in signature verification */
    int dummy; /*dummy variable is used when necessary */
} Msg;
```

Instructions:

1. Write your client.c program in CLIENT directory. Run the program as: ./a.out 127.0.0.1 (for local host loop back)

2. Write your server.c program in SERVER directory. Run the program as: ./a.out

3. You may use existing hash function code (for example, MD5 / SHA-1) from open source codes. Note that your programs should contain at least two hash functions MD5 and SHA-1 implementation. Depending on the choice made for hash function, your program will execute on that chosen function function.

4. Prime number must be chosen very large (for example > 10000) using the Miller Robin Test.

5. For modular inversion, you use the Extended Euclid's GCD algorithm.

6. For modular exponentiation, you use the repeated square-and-multiply algorithm.

7. Display your all calculations at both Alice and Bob sides.

8. Every student is requested to register in course portal for submissions.