Pimpri Chinchwad College of Engineering
Computer Laboratory - III

Assignment No - 4

1 Aim

Produce a DSA signature using parameter tuple {p,q,g}, long term key pair and a message digest

2 Objective

- To understand working of Digital Signature Algorithm generator.
- To generate public and private key for signing message
- To generate message digest from source message
- To sign the message digest using private key
- To verify the authenticity of digital signature using public key

3 Software Requirements

- Linux Operating System
- C++

4 Mathematical Model

S = s, e, x, y, Fme, DD, NDD

S = Initial State
E = End State
X = Input Value
X={x1,x2, x3}
where, x1=p
x2=q
x3=plaintext message
Y =Output
Y={y1,y2}
y1={"Public Key"}
y2={"Private Key"}
Fme = Main function
Fme={f1}
f1="DSA Algorithm to perform data encryption".
Pu=DSA(x3)
Pr=DSA(x3)
where, Pu = Pubic Key.
Pr = Private Key.
DD = Deterministic data {plaintext}
NDD = Non Deterministic Data {public key, private key}

5 Theory

A digital signature algorithm (DSA) refers to a standard for digital signatures. It was introduced in 1991 by the National Institute of Standards and Technology (NIST) as a better method of creating digital signatures. Along with RSA, DSA is considered one of the most preferred digital signature algorithms used today.

Unlike DSA, most digital signature types are generated by signing message digests with the private key of the originator. This creates a digital thumbprint of the data. Since just the message digest is signed, the signature is generally much smaller compared to the data that was signed. As a result, digital signatures impose less load on processors at the time of signing execution, use small volumes of bandwidth, and generate small volumes of

ciphertext intended for cryptanalysis.

DSA, on the other hand, does not encrypt message digests using private key or decrypt message digests using public key. Instead, it uses unique mathematical functions to create a digital signature consisting of two 160-bit numbers, which are originated from the message digests and the private key. DSAs make use of the public key for authenticating the signature, but the authentication process is more complicated when compared with RSA.

The digital signature procedures for RSA and DSA are usually regarded as being equal in strength. Because DSAs are exclusively used for digital signatures and make no provisions for encrypting data, it is typically not subject to import or export restrictions, which are often enforced on RSA cryptography.

6 Algorithm

Parame Generation: The first part of the DSA algorithm is the public key and private key generation, which can be described as:

1. Choose a prime number q, which is called the prime divisor.
2. Choose another primer number p, such that p-1 mod q = 0. p is called the prime modulus.
3. Choose an integer g, such that 1 < g < p, g^q mod p = 1 and g = $h^{(p1)/q} modp$. q is also called g's multiplicative order modulo p.
4. Choose an integer, such that 0 < x < q. Compute y as g^x mod p.
5. Package the public key as p,q,g,y. Package the private key as p,q,g,x.

Signature Generation and Verification: The second part of the DSA algorithm is the signature generation and signature verification, which can be described as:

To **generate** a message signature, the sender can follow these steps:

1. Generate the message digest h, using a hash algorithm like SHA1.

Example 1
The parameters p, q, g are public to all.
q = 71 = a prime number
p = 18 * 71 + 1 = 1279 = prime number
$g = 3^{18} \bmod 1279 = 1157$
We take the message m = 123 and we chose randomly x = 10 for this signature.
$10^{-1} \bmod 71 = 64$
Alice's private key: $X_A = 15$
Alice's public key: $Y_A = 1157^{15} \pmod{1279} = 851$
$r = g^x \pmod{p} \pmod{q} = 1157^{10} \pmod{1279} \pmod{71} = 32$
$s = \frac{x+(r*X_A)}{x} \bmod q = (123 + 15 * 32) * 64 \bmod 71 = 39$
We obtained Alice's signature on the message m = 123, the pair (r, s) = (32, 39).
Verification for the example of DSA

Figure 1: Example 1

2. Generate a random number k, such that 0 < k < q.

3. Compute r as (g^k mod p) mod q. If r = 0, select a different k.

4. Compute i, such that k*i mod q = 1. i is called the modular multiplicative inverse of k modulo q.

5. Compute s = i*(h+r*x) mod q. If s = 0, select a different k.

6. Package the digital signature as r,s.

To find multiplicative inverse:

```
ExtEuclid (a,b) {
// returns a triple (d,s,t) such that d = gcd(a,b) and
// d == a*s + b*t

if (b == 0) return (a,1,0) ;

(d1, s1, t1) = ExtEuclid(b,a%b) ;
```

```
d = d1 ;
s = t1 ;
t = s1 - (a div b) * t1 ;   // note: div = integer division
return (d,s,t) ;
 }
```

To **verify** a message signature, the receiver of the message and the digital signature can follow these steps:

1. Generate the message digest h, using the same hash algorithm.
2. Compute w, such that s*w mod q = 1. w is called the modular multiplicative inverse of s modulo q.
3. Compute u1 = h*w mod q.
4. Compute u2 = r*w mod q.
5. Compute v = (((g^u1)*(y^u2)) mod p) mod q.
6. If v == r, the digital signature is valid.

7 Conclusion

Thus, we have studied and implemented DSA signature using parameter tuple {p,q,g}, long term key pair and a message digest.

Roll No.	Name of Student	Date of Performance	Date of Submission
302	Abhinav Bakshi	04/02/2016	18/02/2016

8 Plagarism Report

Completed: 100% Checked — 63% Unique

To \textbf{verify} a message signature, the receiver of	- Unique
\begin{codeblock} \begin{enumerate} \item Generate the message	- Plagiarized
such that s*w mod q = 1. w is called the modular multiplicative	- Plagiarized
Compute u2 = r*w mod q. \item Compute v = (((g^{u1})*(y^{u2}))	- Unique
valid. \end{enumerate} \end{codeblock} \section{Conclusion}	- Unique
using parameter tuple \{p,q,g\}, long term key pair and	- Plagiarized
\\\ S = Initial State \\ E = End State \\ X = Input Value\\	- Unique
Y =Output\\ Y=\{y1,y2\}\\ y1=\{"Public Key"\}\\ y2=\{"Private	- Plagiarized
to perform data encryption".\\ Pu=DSA(x3)\\ Pr=DSA(x3)\\	- Unique
data \{plaintext\}\\ NDD = Non Deterministic Data \{public	- Unique

9 Output

```
-------------------------------------------
DSA output

Generation
Parameters :
p : 1279
q : 71
g : 1157

Signing
r : 32
s : 39

Verification

*****Signature Verified*****

*****Signature Verified*****

-------------------------------------------
```