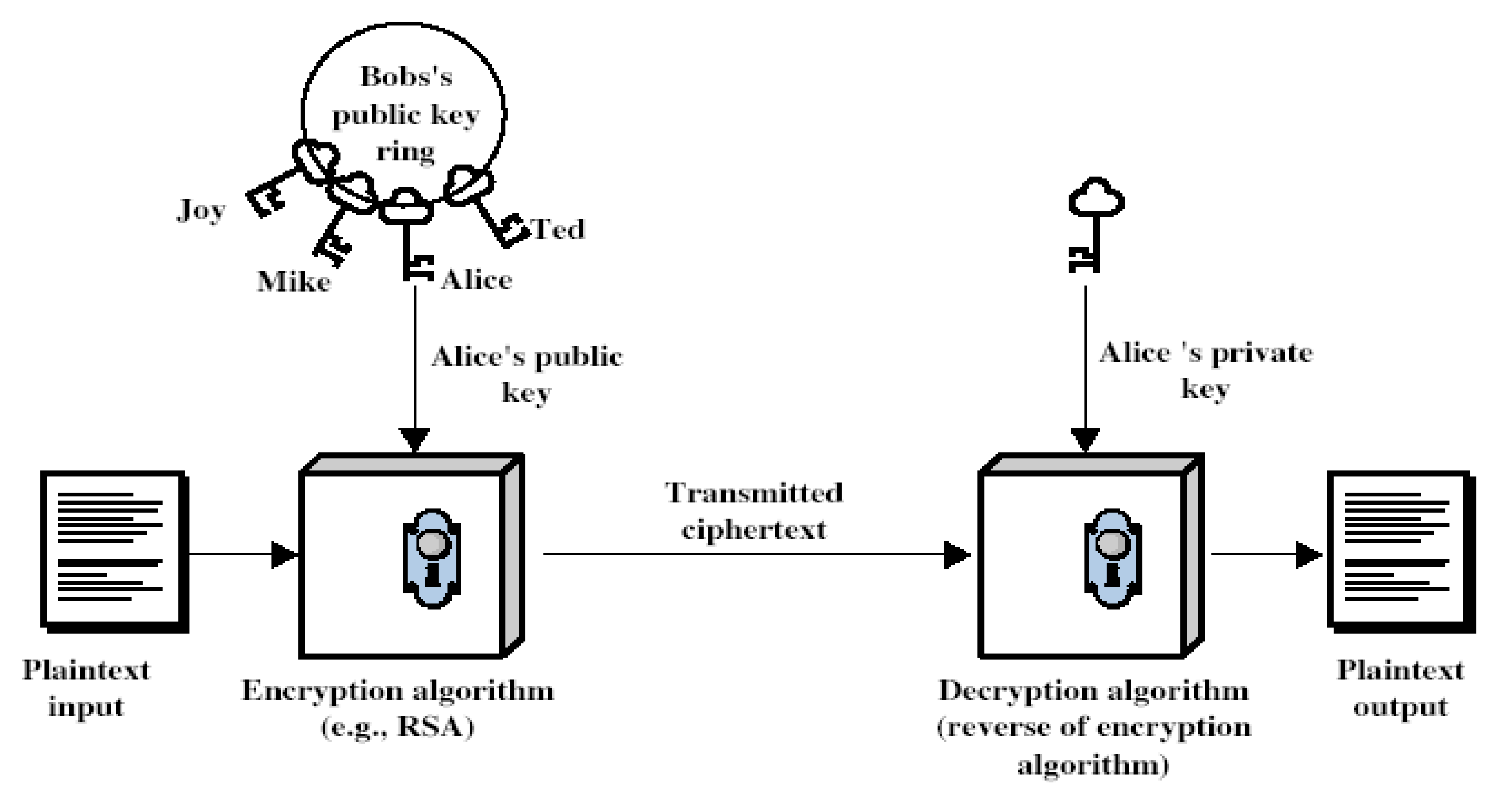
RSA Algorithm

RSA :

- RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described it in 1978.
- **RSA** is an algorithm used by modern computers to encrypt and decrypt messages.
- It is an Asymmetric Cryptographic Algorithm. Asymmetric means that there are two different keys. This is also called *public key cryptography* .
- A user of RSA creates and then publishes the product of two large prime numbers, along with an auxiliary value, as their public key.
- Anyone can use the public key to encrypt a message but only someone with knowledge of the prime factors can feasibly decode the message.

Public-Key Cryptography



Algorithm

- The RSA algorithm involves four steps: key generation, key distribution, encryption and decryption.

1. Choose two distinct prime numbers *p* and *q*.
2. Compute $n = pq$.
3. Compute $z = (p-1)(q-1)$.
4. Choose an integer e, $1 < e < z$, such that $GCD(e, z) = 1$
5. Compute the secret exponent d, $1 < d < z$, such that

   $e \times d \equiv 1 \pmod{z}$

1. The public key is (n, e) and the private key is (n, d).
2. Encrypting messages : $c = m^e \bmod n$
3. Decrypting messages :$m = c^d \bmod n$

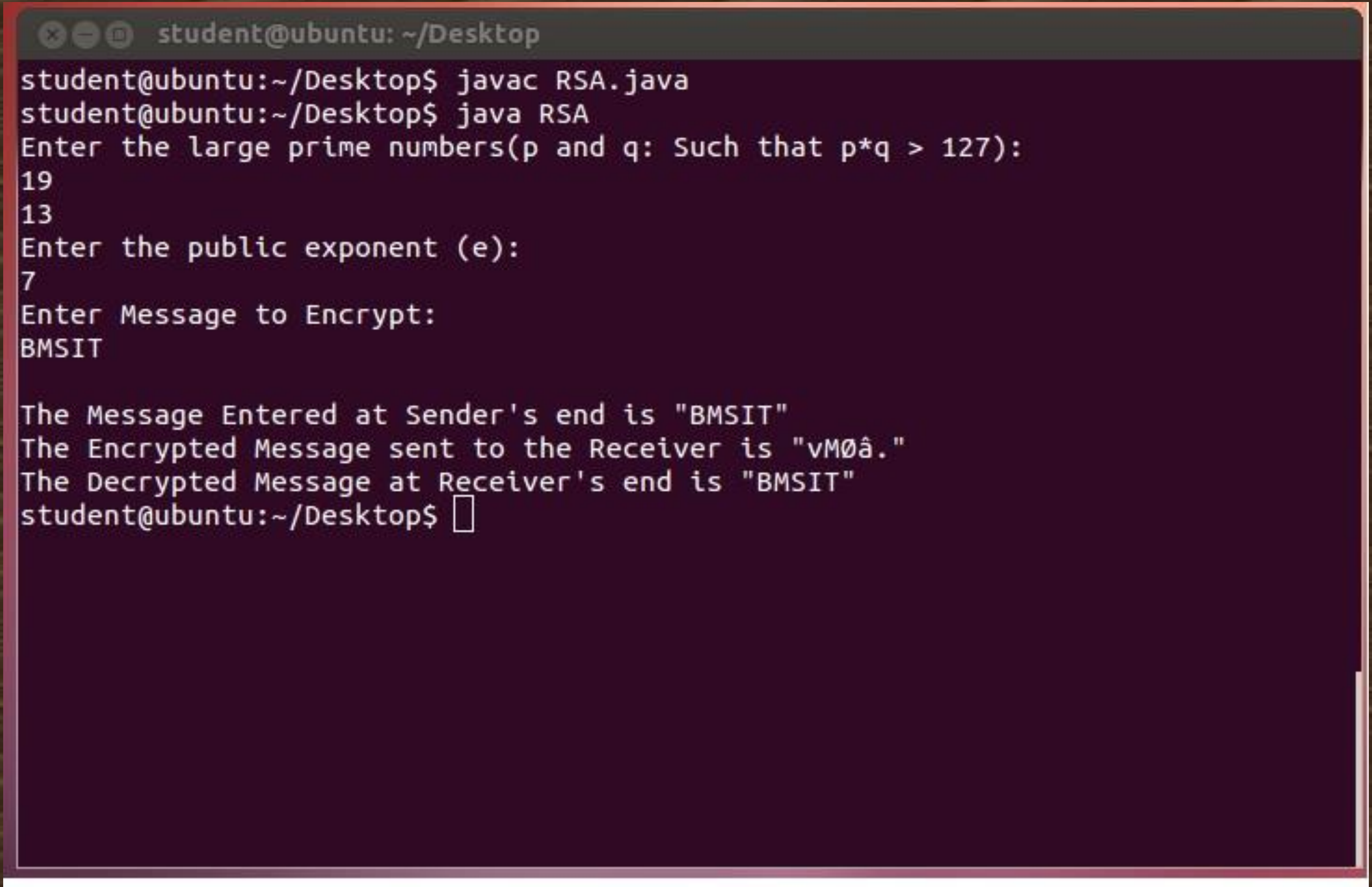
Source Code

```
import java.math.BigInteger;
import java.io.*;
public class RSA
{
BigInteger p, q, d, e, n, z;
BufferedReader keyin = new BufferedReader(new InputStreamReader(System.in));
String msg, rmsg, code;
int size;
BigInteger m, c;
void read()throws IOException
{
System.out.println("Enter the large prime numbers(p and q: Such that p*q > 127):");
p = new BigInteger(keyin.readLine());
q = new BigInteger(keyin.readLine());
n = p.multiply(q); // n = p*q
z=(p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));
System.out.println("Enter the public exponent (e):");
e = new BigInteger(keyin.readLine());
d = new BigInteger("0");
BigInteger temp
do
{
d = d.add(BigInteger.ONE);
temp = (d.multiply(e)).mod(z);
}while(!temp.equals(BigInteger.ONE));
System.out.println("Enter Message to Encrypt:");
msg = keyin.readLine();
size = msg.length();
code = "";
rmsg = "";
}
```

```
void encrypt()
{
for(int i = 0; i < size; i++)
{
m = BigInteger.valueOf((int)msg.charAt(i));
c = m.modPow(e, n);
code += (char)c.intValue();
}
}
void  decrypt()
{
for(int i = 0; i < size; i++)
{
c = BigInteger.valueOf((int)code.charAt(i));
m = c.modPow(d, n);
rmsg += (char) m.intValue();
}
}
```

```
void show()
{
System.out.println("\nThe Message Entered at
    Sender's end is \"" + msg + "\"");
System.out.println("The Encrypted Message
    sent to the Receiver is \"" + code + "\"");
System.out.println("The Decrypted Message
    at Receiver's end is \"" + rmsg + "\"");
}
public static void main(String args[])throws
    IOException
{
RSA obj = new RSA();
obj.read();
obj.encrypt();
obj.decrypt();
obj.show();
}
}
```

Output



```
student@ubuntu:~/Desktop$ javac RSA.java
student@ubuntu:~/Desktop$ java RSA
Enter the large prime numbers(p and q: Such that p*q > 127):
19
13
Enter the public exponent (e):
7
Enter Message to Encrypt:
BMSIT

The Message Entered at Sender's end is "BMSIT"
The Encrypted Message sent to the Receiver is "vMØâ."
The Decrypted Message at Receiver's end is "BMSIT"
student@ubuntu:~/Desktop$ 
```

THANK YOU