# Rivest-Shamir-Adleman Encryption Algorithm

Prashanth.S 19MID0020

## Importing the Necessary Libraries

In [1]:
```
import numpy as np
import random
```

## Operational Functions

In [2]:
```
def si(n): return n-1
```

In [3]:
```
def num_check(num1, num2,condition):
    while condition:
        random_num = random.randint(2, (si(num1) * si(num2)) - 1)
        if (np.gcd(random_num, (si(num1) * si(num2))) == 1):
            break
        else:
            continue
    return random_num
```

In [4]:
```
def modulo_multiplicative_inverse(a, m):
    for x in range(1, m):
        if (((a%m) * (x%m)) % m == 1):return x
    return -1
```

In [5]:
```
## {e,n}
public_key = []

## {d,n}
private_key = []

prime_1 = 3
prime_2 = 11

public_key.append(num_check(prime_1, prime_2, True))
public_key.append(prime_1 * prime_2)


modulo_ans = modulo_multiplicative_inverse(public_key[0], (si(prime_1) * si(prime_2)))
private_key.append(modulo_ans)
private_key.append(public_key[1])
```

In [6]:
```
print(public_key)
print(private_key)
```

```
[17, 33]
[13, 33]
```

In [7]:
```
e = public_key[0]
#e = 7
d = private_key[0]
n = public_key[1]
```

# Encryption

In [8]:
```
message=5
if (message < n):
    cipher_text = (message**e % n)
print(cipher_text)
```

```
14
```

# Decryption

In [9]:
```
decrypt_text = (cipher_text**d % n)
decrypt_text
```

Out[9]: 5

In [10]:
```
if (message == decrypt_text):
    print("Successful Transmission")
else:
    print("Not Successful Transmission")
```

```
Successful Transmission
```