This is a short overview of RSA and how it works, with an example at the end.

Intro:
RSA is based on prime factorization and is an example of public key cryptography, also known as a-symmetric cryptography.

It needs a public key and a private key. The public key is sent to others. Those *others* will encrypt a message 'plain-text', with the received public key to generate 'cipher-text'. The cipher-text is sent to the original owner of the public key. That owner is the only one who is able to decrypt the cipher-text with his private key.

The reason why the owner of the private key is the only one who can decrypt messages that were encrypted with his private key, is that the two keys are mathematically related. We will see that in a second.

```
The public  key contains two numbers: (e, n)
The private key contains two numbers: (d, n)
```

'n' is the product of two prime numbers
'e' is a chosen number (usually **65537**=2^16+1)
'd' is the modular multiplicative inverse of 'e'.

We also need Euler's totient function: Phi. This is the product of (p-1)*(q-1). We need phi to calculate 'd'.

'e'
Should be a prime number between 1 and 'phi', and it should also be coprime with 'phi'.

'd'
Modular multiplicative inverse of 'e'.
So: e*d=1(mod phi)

Choose p and q:
(these are very large primes, we use small ones).

```
p=11 q=5
n=p*q -> 55=11*5
```

Compute phi:

```
phi=(p-1)*(q-1)    40=10*4
```

Choose e:
e=7
is 'e' between 1 and 'phi'? yes
is 'e' coprime with 'phi'?
Euclidean algo to find greatest common divisor (gcd).

```
40=5(7)+5
7=1(5)+2
5=2(2)+1

1 is the greatest common divisor
so 7 is coprime with 40.
```

Find 'd':
e*d=1(mod phi)
7*?=1(mod 40) -> extended euclidean algo

```
5-2(2)=1
5-2(7-1(5))=1
3(5)-2(7)=1
3(40-5(5))-2(7)=1
3(40)-17(7)=1

-17(mod 40)=23
(We need a positive value.)

modular multiplicative inverse: 7*23=1(mod 40)
```

An example:

The message to be encrypted is '8'.

The keys:
Public (e,n)   (7,55)
Private (d,n)   (23,55)

Plain-text = 8

(encrypt)
cipher = *8* to the power of *e* mod *n*
8^7%55 = **2**

Cipher-text = 2

(decrypt)
plain-text = *2* to the power of *d* mod *n*
2^23%55 = **8**