RSA(**R**ivest-**S**hamir-**A**dleman) Algorithm is an **asymmetric**or **public-key cryptography**algorithm which means it works on two different keys: **Public Key**and **Private Key**. The Public Key is used for **encryption**and is known to everyone, while the Private Key is used for **decryption**and must be kept secret by the receiver. RSA Algorithm is named after Ron **R**ivest, Adi **S**hamir and Leonard **A**dleman, who published the algorithm in 1977.

**Example of Asymmetric Cryptography:**

If Person **A** wants to send a message securely to Person **B**:

* Person A **encrypts**the message using Person B’s **Public**Key.
* Person B **decrypts**the message using their **Private**Key.

### RSA Algorithm

*RSA Algorithm is based on* ***factorization*** *of large number and* ***modular arithmetic*** *for encrypting and decrypting data. It consists of three main stages:*

1. ***Key Generation:*** *Creating Public and Private Keys*
2. ***Encryption:*** *Sender encrypts the data using Public Key to get* ***cipher text****.*
3. ***Decryption:*** *Decrypting the* ***cipher text*** *using Private Key to get the original data.*

**1. Key Generation**

* Choose two large prime numbers, say **p** and **q**. These prime numbers should be kept secret.
* Calculate the product of primes, **n = p \* q**. This product is part of the public as well as the private key.
* Calculate [Euler Totient Function](https://www.geeksforgeeks.org/eulers-totient-function/" \t "https://www.geeksforgeeks.org/rsa-algorithm-cryptography/_blank)**Φ(n)**as **Φ(n)** =**Φ(p \* q) = Φ(p) \* Φ(q) = (p – 1) \* (q – 1).**
* Choose encryption exponent **e**, such that
  + 1 < e < Φ(n), and
  + gcd(e, Φ(n)) = 1, that is e should be co-prime with Φ(n).
* Calculate decryption exponent **d,**such that
  + **(d \* e) ≡ 1 mod Φ(n)**, that is d is [modular multiplicative inverse](https://www.geeksforgeeks.org/multiplicative-inverse-under-modulo-m/" \t "https://www.geeksforgeeks.org/rsa-algorithm-cryptography/_blank) of **e**mod Φ(n). Some common methods to calculate multiplicative inverse are: [Extended Euclidean Algorithm](https://www.geeksforgeeks.org/euclidean-algorithms-basic-and-extended/" \t "https://www.geeksforgeeks.org/rsa-algorithm-cryptography/_blank), [Fermat’s Little Theorem](https://www.geeksforgeeks.org/fermats-little-theorem/" \t "https://www.geeksforgeeks.org/rsa-algorithm-cryptography/_blank), etc.
  + We can have multiple values of d satisfying **(d \* e) ≡ 1 mod Φ(n)** but it does not matter which value we choose as all of them are valid keys and will result into same message on decryption.

Finally, the **Public Key = (n, e)** and the **Private Key = (n, d)**.

**2. Encryption**

To encrypt a message **M**, it is first converted to numerical representation using ASCII and other encoding schemes. Now, use the public key (n, e) to encrypt the message and get the cipher text using the formula:

***C = Me mod n****, where C is the Cipher text and e and n are parts of public key.*

**3. Decryption**

To decrypt the cipher text **C**, use the private key (n, d) and get the original data using the formula:

***M = Cd mod n,*** *where M is the message and d and n are parts of private key.*

Key Generation:

1. **Choose two prime numbers:** Let's use p = 3 and q = 11.
2. **Calculate n:** n = p \* q = 3 \* 11 = 33. This is the modulus.
3. **Calculate φ(n):** φ(n) = (p - 1) \* (q - 1) = (3 - 1) \* (11 - 1) = 2 \* 10 = 20.
4. **Choose e:** Select e such that 1 < e < φ(n) and e and φ(n) are coprime (have no common divisors). Let's choose e = 7. (7 and 20 are coprime).
5. **Calculate d:** Find d such that (d \* e) mod φ(n) = 1. One solution is d = 3 because (3 \* 7) mod 20 = 1.

Public Key: The public key is (e, n) = (7, 33).  
Private Key: The private key is (d, n) = (3, 33).

Encryption:

1. **Message (m):** Let's say the message is m = 2.
2. **Ciphertext (c):** c = m^e mod n = 2^7 mod 33 = 128 mod 33 = 29.

Decryption:

1. **Ciphertext (c):** c = 29.
2. **Plaintext (m):** m = c^d mod n = 29^3 mod 33 = 29 \* 29 \* 29 mod 33 = 24389 mod 33 = 2.