# 1 RSA Algorithm

## 1.1 Introduction

This algorithm is based on the difficulty of factorizing large numbers that have 2 and only 2 factors (Prime numbers). The system works on a public and private key system. The public key is made available to everyone. With this key a user can encrypt data but cannot decrypt it, the only person who can decrypt it is the one who possesses the private key. It is theoretically possible but extremely difficult to generate the private key from the public key, this makes the RSA algorithm a very popular choice in data encryption.

## 1.2 Algorithm

First of all, two large distinct prime numbers $p$ and $q$ must be generated. The product of these, we call $n$ is a component of the public key. It must be large enough such that the numbers $p$ and $q$ cannot be extracted from it - 512 bits at least i.e. numbers greater than $10^{154}$. We then generate the encryption key $e$ which must be co-prime to the number $m = \varphi(n) = (p-1)(q-1)$. We then create the decryption key $d$ such that $de \, mod \, m = 1$. We now have both the public and private keys.

## 1.3 Encryption

We let $y = E(x)$ be the encryption function where $x$ is an integer and $y$ is the encrypted form of $x$
$y = x^e \, mod \, n$

## 1.4 Decryption

We let $X = D(y)$ be the decryption function where $y$ is an encrypted integer and $X$ is the decrypted form of y
$X = y^d \, mod \, n$

## 1.5 Simple Example

1. We start by selecting primes $p = 3$ and $q = 11$.

2. $n = pq = 33$
   $m = (p-1)(q-1) = (2)(10) = 20.$

3. Try $e = 3$
   $gcd(3, 20) = 1$
   $\Rightarrow e$ is co-prime to $n$

4. Find $d$ such that $1 \equiv de \, mod \, m$
   $\Rightarrow 1 = Km + de$
   Using the extended Euclid Algorithm we see that $1 = -1(20) + 7(3)$
   $\Rightarrow d = 7$

5. Now let's say that we want to encrypt the number $x = 9$:
   We use the Encryption function $y = x^e \, mod \, n$
   $y = 9^3 \, mod \, 33$
   $y = 729 \, mod \, 33 \equiv 3$
   $\Rightarrow y = 3$

6. To decrypt $y$ we use the function $X = y^d \, mod \, n$
   $X = 3^7 \, mod \, 33$
   $X = 2187 \, mod \, 33 \equiv 9$
   $\Rightarrow X = 9 = x$
   $\Rightarrow$ It Works!

# 2 Inplementing RSA Using Java

The first task is to generate the prime numbers p and q. This is done using the BigInteger class in java. We must use BigInteger instead of the standard int because an integer variable cannot exceed $2^{31} - 1$ while a BigInteger can simulate arbitrary-precision integers. We can apply all the usual mathematical operations to BigInteger as well as others like mod ular arithmetic, gcd, primality testing etc. When constructing a BigInteger we specify a bit-length and the amount of times $t$, that we want the Miller-Rabin probabilistic test (Below) to run on the BigInteger, as well as supplying a random set of bits for these tests. This will generate a random integer which is probably prime with the specified bit-length. The probability that the new BigInteger represents a prime number will exceed $(1 - 1/4^t)$.

From this we can easily generate $n$ and $m$. The next step is to calculate $e$ which must be co-prime to $m$, i.e. $gcd(e, m) = 1$. We begin by letting $e = 3$, if gcd$(e, m) \neq 1$ we let $e$ be the next odd number. We continue in this fashion until the $gcd(e, m) = 1$. The reason we only use odd numbers is because $m$ will always be even so therefore no even number will be co-prime to $m$. The BigInteger class utilizes Euclid's algorithm (Below) to calculate gcd's. We now have all the components of the public key.

We must now calculate $d$ such that $de\, mod\, m = 1$. BigInteger uses the method mod Inverse to find $d$.
$de\, mod\, m = 1$
$\Rightarrow 1 - de = mk$............where k is an integer
$\Rightarrow 1 = mK + de$...........$m$ & $e$ are known.
This has a unique solution because $m$ and $e$ are co-prime - We made it so in the last paragraph. This solution is got using the *Extended Euclid Algorithm* (Below).

We now have all the information we require to encrypt integers. We use the encryption function $f(x) = y = x^e\, mod\, n$. BigInteger uses the method mod Pow to calculate $y$.

## 2.1 Euclid's Algorithm - Greatest Common Divisor

```
Public int gcd (int a, int b)
{
        if (b = 0) return a;
        else return gcd(b,a%b)
}
```

Example 1:
1. gcd(90,48)
   $b \neq 0 \Rightarrow$ gcd(48, 90 mod 48)
2. gcd(48,42)
   $b \neq 0 \Rightarrow$ gcd(42, 48 mod 42)
3. gcd(42,6)
   $b \neq 0 \Rightarrow$ gcd(6, 42 mod 6)
4. gcd(6,0)
   $b = 0 \Rightarrow$ gcd(90,48) = 6

## 2.2 Extended Euclid's Algorithm

```
Public int[] EE (int a, int b, int c, int d, int e, int f)
{
        if (b = 0)
        {
                int [] ret = {0,0,0};
                ret [0] = a; // gcd(a,b)
                ret [1] = e; // coefficient of a
                ret [2] = f; // coefficient of b
                return ret;
```

```
        }
        else
        {
                return EE(b, a%b, e-(a/b)*c, f-(a/b)*d, c, d);
        }
}
// N.B
// c and f must be initialized to 0 for algorithm to work
// d and e must be initialized to 1 for algorithm to work
```

Example 2:

We want to know what the gcd of 108 and 5 is and also we want to find the integers x and y that satisfy 108x + 5y = gcd.

1. EE(108, 5, 0, 1, 1, 0)
   $b \neq 0 \Rightarrow$ EE(5, 108%5, 1-0, 0-21, 0, 1)
2. EE(5, 3, 1, -21, 0, 1)
   $b \neq 0 \Rightarrow$ EE(3, 5%3, 0-1, 1-(-21), 1, -21)
3. EE(3, 2, -1, 22, 1, -21)
   $b \neq 0 \Rightarrow$ EE(2, 3%2, 1-(-1), -21-22, -1, 22)
4. EE(2, 1, 2, -43, -1, 22)
   $b \neq 0 \Rightarrow$ EE(1, 2%1, -1-4, 22-(-86), -1, 22)
5. EE(1, 0, -5, 108, 2, -43)
   $b = 0 \Rightarrow$ ret = {1, 2, -43}

Therefore 1 = 2(108) - 43(5)

## 2.3 The Miller-Rabin Probabilistic Test

Given an integer x, we want to test in for primality we can apply the Miller-Rabin probabilistic test. The algorithm is as follows:

1. A random number $b$ is chosen from the set of integers $[1, (n-1)]$
2. We must find $q$ and the odd number $m$ such that $n - 1 = 2^q m$.
3. We then test if either of the following conditions hold:

   **(a)** $b^m \, mod \, x \equiv 1$ OR

   **(b)** If $\exists$ an integer $i \in [0, (q-1)]$ such that $-1 \equiv b^{m2^i} \, mod \, x$

4. If neither of the above conditions are satisfied
   $\Rightarrow x$ is definitely composite.
   However if either (a) or (b) are true
   $\Rightarrow x$ is possibly prime(Inconclusive).

If we conduct k of these tests and all k tests are inconclusive
$\Rightarrow$ The probability of $x$ being prime is $(1-(\frac{1}{4})^k)$.

However if any of these test fail
$\Rightarrow x$ is composite

### 2.3.1 Java Code For M-R Probabilistic test

```
import java.util.Random;
import java.math.BigInteger;
Public class mrpt
{
        public int primeT(int p)
        {
                Random gen = new Random();
                int b = gen.nextInt(p-1)+1;
                int [] qandm = getqm(p);
                int q =qandm[0];
                int m =qandm[1];
                BigInteger bval = new BigInteger((""+b));
                BigInteger mval = new BigInteger((""+m));
                BigInteger qval = new BigInteger((""+q));
                BigInteger pval = new BigInteger((""+p));
                BigInteger two = new BigInteger("2");
                BigInteger pminusone = new BigInteger(""+(p-1));

                if (q==-1)return 0;
                if (bval. modPow(mval,pval).compareTo(BigInteger.ONE)==0)return 1;
                int j = 0;
                BigInteger indexval = mval;
                while (j < q)
                {
                        if (pminusone.compareTo(bval. modPow(indexval,pval))==0)return 1;
                        indexval = indexval.multiply(two);
                        j++;
```

```
            }
            return 0;
        }

        public int [] getqm(int p)
        {
            p = p-1;
            int [] rt ={0,0}; // rt = {q, m}
            if (p%2 != 0)
            {
                rt[0] = -1; rt[1] = -1;
                return rt;
            }
            int divisor = p/2;
            int count = 1;
            double maxq = (Math.log(p))/(Math.log(2));
            while (count <= maxq && divisor%2==0)
            {
                count++;
                divisor = divisor/2;
            }
            rt[0] = count; rt[1] = divisor;
            return rt;
            }
        }
}
```

Example 3:

We want to know if the integer $x = 15$ is prime.

1. $B$ is chosen at random...lets say $B = 8$.
2. We then solve (15 - 1) = $2^q m$
   $\Rightarrow m = 7$ and $q = 1$
3. **(a)** Is $8^7 \, mod \, 15 \equiv 1$ ?
   $2097152 \, mod \, 15 \equiv 2$
   $\Rightarrow$ false

(b) Does an integer $i \in [0,(q-1)]$ exist such that
$-1 = b^{m2^i} \, mod \, x$.
In this case i = 0 is the only possibility but from
(a) we can see that $8^{7.2^0} \, mod \, 15 \equiv -1$ is false

$\Rightarrow x$ is composite.

Example 4:

We want to know if the integer $x = 17$ is prime.

1. $B$ is chosen at random...lets say $B = 3$.
2. We then solve (17 - 1) = $2^q m$
$\Rightarrow m = 1$ and $q = 4$
3. **(a)** Is $3^1 \, mod \, 17 \equiv 1$ ?
$\Rightarrow$ false

   **(b)** Does an integer $i \in [0,(q-1)]$ exist such that
$-1 = b^{m2^i} \, mod \, x$.
for i = 0 : $3^{1.2^0} \, mod \, 17 \equiv 3$
for i = 1 : $3^{1.2^1} \, mod \, 17 \equiv 9$
for i = 2 : $3^{1.2^2} \, mod \, 17 \equiv 13$
for i = 3 : $3^{1.2^3} \, mod \, 17 \equiv -1$
$\Rightarrow$ true

$\Rightarrow x$ is possibly prime

# 3 Mathematics Of The RSA Algorithm

Given: n = pq where p and q are distinct primes.
$gcd(e, \varphi(n)) = 1$
$de = 1 \, mod \, \varphi(n)$
When $y = x^e \, mod \, n$ and $X = y^d \, mod \, n$
where $x < min\{p, q\}$
Prove that : $X = x \, mod \, n \; \forall \, x < n$

Proof: $X = x^{de} \, mod \, n$
$de = 1 \, mod \, \varphi(n)$
$\varphi(n) = (p-1)(q-1)$ if p and q are distinct primes
$de = 1 + k(p-1)(q-1)$

$X = x^{1+k(p-1)(q-1)}$
$X = x.(x^{(p-1)})^{k(q-1)}$

But $x^{(p-1)} = x^{\varphi(p)}$ *and* $x \in Z_p^*$
So $x^{(p-1)} = 1 \, mod \, p$ ...Fermat/Euler Theorem
So $X = x.(1 \, mod \, p)^{k(q-1)}$
So $X = x \, mod \, p$
Similarly $X = x \, mod \, q$
Because p and q are co-prime we can use the Chinese remainder Theorem
Therefore $X = x \, mod \, pq$
$\Rightarrow X = x \, mod \, n$

## 3.1 Fermat/Euler Theorem

Theorem $\forall \, x \in Z_n^*, \; x^{\varphi(n)} \equiv 1 \, mod \, n$

Proof $Z_n = \{1, 2...(n-1)\} \, mod \, n$
$Z_n^* = \{x \, \in Z_n : gcd(x, n) = 1\}$
The order of $Z_n^*$ *is* $\varphi(n)$ and is called the *Euler Function*

We let $u_1, ...., u_{\varphi(n)}$ be an enumeration of all the elements of $Z_n^*$.
It is clear that $x.u_1, ...., x.u_{\varphi(n)}$ is also an enumeration of all the elements of $Z_n^*$.
Therefore $x.u_1...x.u_{\varphi(n)} = u_1...u_{\varphi(n)}$
So $x^{\varphi(n)}.u_1...u_{\varphi(n)} = u_1...u_{\varphi(n)}$
We let $g = u_1...u_{\varphi(n)}$
$g \in Z_n^* \Rightarrow g^{-1} \in Z_n^*$
So $x^{\varphi(n)}.u_1...u_{\varphi(n)}.g^{-1} = u_1...u_{\varphi(n)}.g^{-1}$
So $x^{\varphi(n)} = 1 \, mod \, n$

## 3.2 Chinese Remainder Theorem

Theorem $x = y \, mod \, p$
$x = y \, mod \, q$
$\Rightarrow x = y \, mod \, pq$

Proof $x = y \, mod \, p$
$\Rightarrow p|(x-y)$
$x = y \, mod \, q$
$\Rightarrow q|(x-y)$
$p$ and $q$ are co-prime
$\Rightarrow pq|(x-y)$

$\Rightarrow x = y\, mod\, pq$

## 3.3 Questions

1. Why must $p$ and $q$ be distinct?

   If they are the same the above algorithm will fail. This is due to the fact that $\varphi(n) = (p-1)(q-1)$ if and only if $p$ and $q$ are distinct. However if $p = q \Rightarrow \varphi(n) = (p)(p-1)$

2. Why must $x < min\{p, q\}$?

   Well, one step in the proof of RSA uses the Fermat/Euler Theorem, to establish that $x^{p-1} = 1(modp)$ For this to work, $x$ must not equal $p$. For the whole algorithm to work, $x$ must also not equal $q$. So, while it generally works for $x < N$, if you land on $p$ or $q$ by chance then it will fail. To be on the safe side, it's usually said that $x < min\{p, q\}$.