# RSA

# Prime Numbers

- An integer $p$ is a **prime number** if it has no factors other than $1$ and itself.
- An integer which is greater than $1$ and not a prime number is said to be **composite.**
- Thus given a composite number $c$ we know that $c=r*s$ for some non-trivial integers $r$ and $s$.

# Factorisation

- Given an integer *n*, there is an efficient algorithm to determine whether n is composite or prime.
- However determining the factors of a large composite number is a very hard problem.
- Known as the ***factorisation problem*** – this is the basis of the RSA cryptosystem.

- The fastest factorisation algorithm at the moment is called the "*Number Field Sieve*" but even this is not all that efficient.
- To find the factors of a composite number $n$ which is the product of 2 large primes, and has about 640 binary bits (approximately 200 decimal digits) is an impossible task even if you could use all of the computing power in the world!

# Important to Note:

1. Determining whether a large number is prime or composite is easy;
2. Multiplying 2 large numbers together is easy;
3. Factorising a large number which is the product of 2 large primes (i.e. retrieving the original prime factors) is very difficult.

# Fermat's Little Theorem

If $p$ is a prime number and $a$ is any number between $1$ and $p-1$ inclusive, then

$$\boxed{a^{p-1} \bmod p = 1}$$

This is not true in general, which gives us a method to decide if a given number $n$ is prime or composite.

# Solving a problem

Suppose I have

- a prime number *p*;
- a number *m* between *1* and *p-1* inclusive;
- another number *e* also between *1* and *p-1*;

And I compute

- $c = m^e \; mod \; p$

If I give you *c,e* and *p* can you find *m?*

Yes you can if you take the following steps:

1. Find a number $d$ such that $e*d=1 \bmod p-1$
2. Compute $c^d \bmod p = m$

Why does that work?

1. We found $d$ such that $e*d = 1 \bmod p\text{-}1$

   That means that $e*d - 1 = k(p\text{-}1)$ for some value of $k$.

   Or

   $$\boxed{ed = k(p\text{-}1) + 1}$$

2. We computed $c^d \; mod \; p$

But $c^d = (m^e)^d \; mod \; p$

$= m^{ed} \; mod \; p$

$= m^{k(p-1)+1} \; mod \; p$

$= m^{k(p-1)} * m \; mod \; p$

$= 1 * m \; mod \; p$

$= m \; mod \; p$

- This works because of Fermat's Little Theorem.
- We know that since $p$ is a prime we have $a^{p-1}=1 \mod p$ for all $a$ and so $c^{k(p-1)} = 1 \mod$ p leaving us with the answer $m$ in step 2.

- *BUT* if the modulus is not a prime number then the method doesn't work.

# Why doesn't it work?

- In general $a^{n-1} \neq 1 \ mod \ n$ if $n$ is not prime.

- We could make the method for finding $m$ work if we knew the number $r$ such that

  $$a^r = 1 \ mod \ n$$

  If $a$ and $n$ are co-prime then there will be such a number $r$ and there is a way to find it

# Finding *r*

- In order to find $r$ such that $a^r = 1 \bmod n$, you have to be able to factorise $n$ and find all of its prime factors.
- If $n = p*q$ where $p$ and $q$ are primes then

$$\boldsymbol{r = (p-1)*(q-1)}$$

# Important to note now:

1. It is easy to determine whether a large number is prime or composite.
2. It is easy to compute the product of two large primes $n = p*q$.
3. Setting $r = (p-1)*(q-1)$ we have

   $$m^r = 1 \ mod \ n$$

   for all $m$ co-prime with $n$.

4. Given $e$ (co-prime with $r$), it is easy to determine $d$ such that

   $$(e*d) = 1 \mod r$$

5. It is easy to compute $m^e \mod n$
6. If $c=m^e \mod n$ then $m=c^d \mod n$ and it is easy to compute $c^d \mod n$ if you know $d$.
7. You can only find $d$ if you can find $r$ and you can only find $r$ if you can factorise $n$.
8. Factorising $n$ is hard.

9. This is the basis of the RSA public key cryptosystem. The holder of the public key knows $p$ and $q$ and therefore he/she can find $r$ and therefore $d$ and can compute $c^d \mod n$ to find $m$.
10. No-one else knows $p$ and $q$, so they cannot find $r$ or $d$ and so they cannot recover $m$.
11. There is no known way to recover $m$ which is not equivalent to factorising $n$.

# RSA – Key Generation

1. Bob generates two large primes $p$ and $q$ (each with approx. 100 decimal digits).
2. He computes $n = p*q$
3. He computes $r = (p-1)*(q-1)$
4. He chooses a random number $e$ which is between $1$ and $r$ which has no factor in common with $r$.

5. He computes the **private key** *d* by solving the equation $(e*d) = 1 \bmod r$.
6. He can now carefully dispose of the values of *p, q* and *r*.
7. Bob keeps *d* private but publishes the value of the pair *(e,n)*. This is his **public key.**

# RSA - Encryption

Alice wishes to send Bob a message *m*. She takes the following steps:

1. She looks up Bobs public key pair *(e,n)* .
2. She computes $c=m^e \, mod \, n$ and sends the value of *c* to Bob

# RSA - Decryption

Bob receives the value $c$ from Alice. He decrypts it using his private key $d$ by computing

$$m=c^d \bmod n$$

# Notes

- The message $m$ must be smaller than $n$. Alice breaks her message up into blocks each with a value less than $n$ and encrypts each of these blocks individually.
- The public key can be used by anyone wishing to send Bob a message. He does not need a separate key pair for each correspondent.