RSA Cryptosystem

→ it is a **public key cryptosystem** (so it has a private key and a public key)

→ it was constructed in **1977** by **Rivest**, **Shamir** and **Adleman**

→ every public key cryptosystem relies heavily on a trapdoor function
~ **RSA** is secure because of the integer factorization problem

Integer factorization is a trapdoor function: validating the result by multiplying two numbers is quite easy but finding the factors is hard

RSA Cryptosystem

Let **p** be a prime number then for any integer **a** (**a** is not divisible by **p**) the number

$a^{p-1} - 1$ is an integer multiple of **p**.

$$a^{p-1} \equiv 1 \pmod{p}$$

„Fermat's little theorem"

We can generalize this theorem with Euler's **Φ(n)** function: this totient function counts the positive integers up to a given integer **n** that are relative prime to **n**

$$a^{\Phi(n)} \equiv 1 \pmod{p}$$ if **n** and **a** are relative primes

Relative prime: two integers **a** and **b** are said to be relative prime or coprime if the only positive integer (factor) that divides both of them is **1**

gcd(a,b)=1

RSA Cryptosystem

Φ(5) = 1,2,3,4 → so the value of the function is **4**

Φ(8) = 1,3,5,7 → so the value of the function is **4**

Φ(7) = 1,2,3,4,5,6 → so the value of the function is **6**

A very important feature of Euler's **Φ(n)** function is that it is quite easy to calculate for prime numbers

Φ(prime) = prime-1

~ of course because a prime is comprime by definition with all the smaller integers within the range **[1,prime-1]**

WE CAN USE THIS FEATURE IN THE RSA CRYPTOSYSTEM !!!

RSA Cryptosystem

RSA ALGORITHM

1.) generate **2** large prime numbers **p** and **q**

~ we can use Rabin-Miller algorithm to do so

2.) calculate $\mathbf{n = p * q}$ so let's multiply the prime numbers

$$\Phi(n) = (p-1)(q-1)$$

3.) let's calculate the public key **e** parameter

We can calculate **e** such that $\mathbf{gcd(e,\Phi(n))=1}$

~ so basically **e** and **Φ(n)** are relative primes

e and **Φ(n)** share no other factor than **1**

4.) let's calculate the private key **d** parameter: let's calculate the modular inverse of **e**
(this is why it is crucial that **e** and **Φ(n)** is coprime**)**

$$\mathbf{d * e \bmod \Phi(n) = 1}$$

we have to solve this equation to get the **d** parameter

PUBLIC KEY: (e,n) **PRIVATE KEY: (d,n)**

RSA Cryptosystem

RSA ALGORITHM

PUBLIC KEY: (e,n) **PRIVATE KEY: (d,n)**

→ first we have to transform the plaintext into blocks where every block is smaller than **n**

→ as usual we use the public key for encryption and the private key for decrytion

we can use **ASCII** table
to convert text into numbers

$$\text{ciphertext\_block} = \text{plaintext\_block}^{e} \bmod n$$

$$\text{plaintext\_block} = \text{ciphertext\_block}^{d} \bmod n$$

RSA Cryptosystem

RSA ALGORITHM EXAMPLE

1.) let's generate large prime numbers: **p=17** and **q=23**

2.) let's calculate **n =p ∗q=17x23=391** so **Φ(n)=(17-1)(23-1)=352**

3.) we have to find an **e** number where **gcd(e, Φ (n))=1** so **e=21**

4.) we have to find the modular inverse of **e** so **d=285**

Public key: **(21,391)** Private key: **(285,391)**

For example: we have the character **a** we want to encrypt. The **ASCII** representation of **a** is **97**

Encryption → **ciphertext_block = plaintext_block ^{e} mod n** = **97 ^{21} mod 391 = 37**

Decryption → **plaintext_block = ciphertext_block ^{d} mod n** = **37 ^{285} mod 391 = 97**

RSA Cryptosystem

CRACKING RSA ALGORITHM

→ the attacker has the public key **(e,n)** pair

→ the aim of the attacker is to calculate the private key **(d,n)** pair

n is not a problem because it is public !!!

OK luckily **RSA** algorithm is public so the attacker takes a look at the theoretical background and the implementation as well

d can be calculated if we know **e** and the Euler's **Φ(n)** function

the attacker knows **e** because it is part of the public key

the attacker knows **n** is the multiple of two primes (**p** and **q**)

INTEGER FACTORIZATION TRAPDOOR FUNCTION !!!

RSA Cryptosystem

CRACKING RSA ALGORITHM

→ factoring large numbers is usually hard: but not always

→ if a given number has smaller factors then it may happen that the factors can be found within hundreds or thousands of iterations

So somehow we have to make sure the prime factors will be large ...

This is where prime numbers have been proved to be important: if we have **p** and **q** large prime numbers then we can calculate **n = p*q** quite fast

What are the factors of **n**? Of course the factors are **p** and **q** and we know that these are large primes (this is exactly why we chose them)

THE REASON WHY WE USE PRIME NUMBERS IS TO MAKE SURE FACTORIZATION IS PRACTICALLY IMPOSSIBLE