Math facts

- finding a large prime (large is > 512 bits or 155 digits) is easy
- multiplying 2 large integers is easy
- factoring a large integer is nearly impossible
- modular exponentiation is easy: given n, m and e it's easy to compute $c = m^e \pmod n$
- modular root extraction (the reverse of modular exponentiation) is easy: given c and e and prime factors p and q, it's easy to recover the value m such that $c = m^e \pmod n$. Modular root extraction is otherwise hard
- Euler's totient function (noted ϕ) counts the positive integers up to a given integer n that are relatively prime to n: if p is prime, then $\phi(p) = p - 1$

RSA algorithm

- Generate a pair of large, random primes p and q
- Compute the product $n = pq$ (n is called a semi-prime)
- Select an odd public exponent e between 3 and $(n-1)$ that is coprime to $\phi(n) = (p-1)(q-1)$
- Compute the private exponent d from e, p and q such that d is the multiplicative inverse of e:

$$e.d \equiv 1 \ (mod \ \phi(n))$$

which means, for some integer k: $\quad e.d \ = 1 + k\phi(n)$

- Output (n, e) as the public key and (p, q, d) as the private key

Encrypting plain text M to get cyphertext C

If M is the plaintext: $C \equiv M^e \ (mod\, n)$

Decrypting the cypher text C

- $C^d \equiv (M^e)^d \equiv M^{ed} \equiv M^{1+k\phi(n)} \equiv M.M^{k\phi(n)} \ (mod\, n)$

But applying Euler's theorem:

- $M^{\phi(p)} = M^{p-1} \equiv 1 \ (mod\, p)$
- $M^{\phi(q)} = M^{q-1} \equiv 1 \ (mod\, q)$

and using basic congruence properties:

- $(M^{p-1})^{k(q-1)} = M^{k(p-1)(q-1)} \equiv M^{k\phi(n)} \equiv 1 \ (mod\, p)$
- $(M^{q-1})^{k(p-1)} = M^{k(p-1)(q-1)} \equiv M^{k\phi(n)} \equiv 1 \ (mod\, q)$

from the previous, the Chinese remainder theorem implies: $M^{k\phi(n)} \equiv 1 \quad (mod\, n)$

- therefore: $C^d \equiv M.M^{k\phi(n)} \equiv M \ (mod\, n)$ which recovers M