RSA: brief review of number theory concepts

The math behind RSA involves modular arithmetic and a few other number-theoretic ideas:

Greatest common divisor (gcd): $gcd(a, b)$ is the largest integer factor that divides perfectly into both a and b

Co-prime: a and b are co-prime if $gcd(a, b) = 1$.

Modular inverse: Given a number a and modulus N, the inverse of a is the integer b such that $ab \equiv 1 \bmod N$. This exists *only if a and N are co-prime*.

Fermat's little theorem: if p prime and a is an integer, then $a^p = a \bmod p$. If a is not divisible by p, then $a^{p-1} \equiv 1 \bmod p$.

RSA: key generation

Step 1

Choose two *prime numbers*, p and q.

Step 2

Compute:

$$N = p \cdot q$$
$$\theta = (p-1)(q-1)$$

RSA: key generation

Step 3

Choose a value e that is *co-prime* with θ.

Step 4

Compute the inverse of e mod θ, i.e., find d s.t.

$$d \cdot e \equiv 1 \bmod \theta$$

The *public key* is the pair (e, N).

The *private key* is the pair (d, N).

RSA: the protocol

Encoding

Acquire the public key (e, N) of the party you wish to send something to. To send the message m, encode it as

$$c = m^e \bmod N$$

Decoding

If you receive c and have the private key (d, N), decode like so:

$$c^d \bmod N = (m^e)^d \bmod N = m$$

Two cases to consider to understand why this works.

RSA: why it works

Since $ed = 1 \bmod \theta$, there exists integer k such that $ed = 1 + k\theta$.

Case 1: m co-prime with N

$$\begin{aligned} c^d \bmod N &= (m^e)^d \bmod N \\ &= m^{1+k\theta} \bmod N \\ &= mm^{k\theta} \bmod N \end{aligned}$$

It is a known result in number theory that if m and N are co-prime, then $m^\theta \equiv 1 \bmod N$ where $\theta = (p-1)(q-1)$. Thus,

$$\begin{aligned} c^d \bmod N &= mm^{k\theta} \bmod N \\ &= m \bmod N \end{aligned}$$

RSA: why it works

Case 2: m not co-prime with N

Then, $gcd(m, N) > 1$. Must be p or q, because those are the only two factors of N.

Suppose $gcd(m, N) = p$. Then, p also divides m,

$$m \equiv 0 \bmod p, \quad m^{ed} \equiv 0 \equiv m \bmod p$$

But q does not. q is prime, so $gcd(q, m) = 1$. So by Fermat's little theorem,

$$m^{(q-1)} \equiv 1 \bmod q, \quad m^{(p-1)(q-1)} = m^{\theta} \equiv 1 \bmod q$$

RSA: why it works

Case 2: m not co-prime with N

Again, since $ed = 1 \bmod \theta$, there exists k such that $ed = 1 + k\theta$.

$$\begin{aligned} m^{ed} \bmod q &= m m^{k\theta} \bmod q \\ &= m \bmod q \end{aligned}$$

So we have

$$m^{ed} \equiv m \bmod p$$
$$m^{ed} \equiv m \bmod q$$

It follows that

$$m^{ed} \equiv m \bmod N$$

RSA: how to break it

- To decrypt the message, we must learn d
- We know e, and that $de \equiv 1 \bmod \theta$

But we don't know θ! We have only e, and N.

However, N and θ are based on the same two prime numbers:

$$N = p \cdot q$$
$$\theta = (p - 1)(q - 1)$$

So if we can factor $N = pq$, then we can decode the message!

The security of RSA relies on the fact that factoring for large numbers is computationally intractable.