rsa

Philippe Carphin

April 2020

1 Introduction

RSA encryption is mostly based on these three things:

1. Factoring a large number is way hard
2. Euler's totient function $\phi(n)$, which says how many numbers between 1 and $n-1$ do not share factors with n, is
   - Hard to calculate if you do not know the factors of n
   - Easy to calculate if you know the factors of n.
3. $a^{k*\phi(n)} \equiv 1 \mod n$ (this is known as Fermat's little theorem).

Also sidenote on modular arithmetic in math versus in computer science. In math, we say that $11 \equiv 4 \mod 7$ meaning that we consider them the same mod 7 and we will be considering everything $\mod n$. We will consider two numbers to be equivalent $\mod n$ if they differ by a multiple of n. The following

$$\begin{aligned}(a+kn)\cdot(b+ln) &= ab + aln + bkn + kln^2 &(1)\\ &= ab + (al + bk + kln)n &(2)\\ (a+kn)+(b+ln) &= a + b + (k+l)n &(3)\end{aligned}$$

shows that $ab \equiv (a+kn)\dot{(}b+ln) \mod n$ This means that in modular arithmetic we can replace any of the numbers with another one that is equivalent modulo n and the result will be equivalent modulo n.

$$15 \cdot 71 \equiv 1 \cdot 1 \mod 7 \qquad (4)$$

For example

$$15 \cdot 71 = 1065 \qquad (5)$$

but if we were just interested in the result modulo 7 we need only note that $15 \equiv 1 \mod 7$ and $71 \equiv 1 \mod 7$. So the result $\mod 7$ is just 1.

2 Initial setup and idea

The idea is to take p, q, two large primes then calculate

$$n = pq.$$

For a prime number p, no other numbers than p itself are factors of p, so $\phi(p) = p - 1$. You'll have to trust me that $\phi(mn) = \phi(m)\phi(n)$. We can therefore calculate $\phi(n) = \phi(p)\phi(q) = (p-1)(q-1)$.

All we have left to do is to find two things that when multiplied together make $k * \phi(n) + 1$. Then we will have

$$\begin{aligned} a^{k\phi(n)+1} &\equiv a^{k\phi(n)} \cdot a \mod n && (6)\\ &\equiv 1 \cdot a \mod n && (7)\\ &\equiv a \mod n && (8) \end{aligned}$$

We will find e, d such that $e \cdot d = k\phi(n) + 1$. We will publish e, n so that someone may take their message m, encrypt it by calculating $m^e \text{mod } n$. They will send us this m^e and because we know d, we will be able to get m back:

$$\begin{aligned} (m^e)^d &= m^{ed} && (9)\\ &= m^{k\phi(n)+1} && (10)\\ &\equiv m \mod n && (11) \end{aligned}$$

3 Final setup

We start by trying to find e such that $\gcd(e, \phi(n)) = 1$.

In my implementation, I just try different values for e and check whether $\gcd(e, \phi(n)) = 1$.

It is a theorem in mathematics that for any a, b there exist x, y such that

$$xa + yb = \gcd(a, b).$$

In our case, that means that if we have found an e that has $\gcd(e, \phi(n)) = 1$, then there exists d, k such that

$$de * k\phi(n) = 1.$$

4 Using the algorithm

We start with p, q two large prime numbers. We calculate their product n, then we calculate $\phi(n) = (p-1)(q-1)$ and find e, d, k such that $ed + k\phi(n) = 1$ (we can see this as meaning that ed is one above a multiple of $\phi(n)$, which is the important thing for us).

Let $m < n$ be our message.

Next $m^e \mod n$ is the encrypted message, lets call it w (upside down m).

To decrypt the encrypted message w, and we raise it to the power 'd'

$$\begin{aligned} w^d &\equiv (m^e)^d \mod n && (12)\\ &\equiv m^(ed) \mod n && (13)\\ &\equiv m^(y * phi + 1) \mod n && (14)\\ &\equiv m^(y * phi) * m \mod n && (15)\\ &\equiv 1 * m \mod n && (16)\\ &\equiv m \mod n && (17) \end{aligned}$$