Public Key Cryptography

We will look at some basic number theoretic algorithms leeding to the RSA Public Key Cipher.

1 Raising to Large Powers by Squaring

How do we compute $a^m \pmod{n}$ when m is large? We could multiply a by itself $m-1$ times (doing the calculation mod n so that we don't have to deal with huge numbers). But there is a much more efficient way.

Algorithm to Compute $a^m \pmod{n}$

1) Write m in base 2,

$$m = \sum_{i=0^r} u_i 2^i,$$

where $r = \lfloor \log_2(m) \rfloor$ and $u_i = 0$ or 1 for each i.

2) By succesive squaring calculate $N_i = a^{2^i} \pmod{n}$ for $i = 0, \ldots, r$.

3) Calcuate $a^m = N_0^{u_0} N_1^{u_1} \ldots N_r^{u_r} \pmod{n}$.

Why does this work?

$$a^m = a^{\sum_{i=0}^r u_i 2^i} = \prod_{i=0}^r a^{u_i 2^i} = \prod_{i=0}^r (a^{2^i})^{u_i} \equiv \prod_{i=0}^r N_i^{u_i} \pmod{n}.$$

Note that step 2) takes r multiplications and step 3) takes at most r mulitplications. Thus we are only doing at most $2\log_2(m)$ multiplications, as opposed to $m-1$ multiplications.

For example we compute $5^{337} \pmod{2349}$.
$5^2 \equiv 25 \pmod{2349}$
$5^4 \equiv 625 \pmod{2349}$
$5^8 \equiv 691 \pmod{2349}$
$5^{16} \equiv 634 \pmod{2349}$
$5^{32} \equiv 277 \pmod{2349}$
$5^{64} \equiv 1561 \pmod{2349}$
$5^{128} \equiv 808 \pmod{2349}$
$5^{256} \equiv 2196 \pmod{2349}$

Next write 351 in base 2.

$$337 = 256 + 64 + 16 + 1$$

Thus

$$5^{337} = 5^{256} \cdot 5^{64} \cdot 5^{16} \cdot 5 \equiv (2196)(1561)(634)(5) \equiv 1049 \pmod{(2349)}$$

We could also use the following recursive algorithm

$$a^m \equiv \begin{cases} 1 & \text{if } m = 0 \\ a(a^{m-1}) & \text{if } m \text{ is odd} \\ (a^{m/2})^2 & \text{if } m \text{ is even} \end{cases} \pmod{n}$$

2 Computing kth roots

Suppose we want to solve $x^k \equiv b \pmod{n}$ where
i) We know $\phi(n)$;
ii) $\gcd(k, \phi(n)) = 1$
iii) $\gcd(b, n) = 1$ or $n = pq$ where p and q are distinct primes.

Theorem 2.1 *For k, n, b as above the congruence $X^k \equiv b$ has a solution in $\mathbb{Z}_n$.*

Algortithm to solve $x^k \equiv b \pmod{n}$
1) Find $s, t \geq 0$ solving

$$ks - \phi(n)t = 1$$

2) Compute $x = b^s \pmod{n}$ using the algorithm from §1.

Why does this work?

First note that since $\gcd(k, \phi(n)) = 1$ we can always find s, t as in 2). If $\gcd(b, n) = 1$, then by Euler's Theorem $b^{\phi(n)} \equiv 1 \pmod{n}$. Thus

$$x^k \equiv (b^k)^s \equiv b^{ks} \equiv b^{1+\phi(n)t} \equiv b(b^{\phi(n)})^t \equiv b \pmod{n}.$$

If $\gcd(b, n) > 1$ and $n = pq$ for distinct primes p, q, we need the following lemma.

Lemma 2.2 *If $n = pq$ where p, q are distinct primes, then for all $r \geq 1$ and all b*

$$b^{r\phi(n)+1} \equiv b \pmod{n}.$$

Proof If $\gcd(b, n) = 1$, this follows from Euler's Theroem. If $\gcd(b, n) = n$, then $b^{r\phi(n)+1} \equiv b \equiv 0 \pmod{n}$. So the only interesting case is when b is divisible by one, but not both of p and q. Suppose $b = mp^l$ where $\gcd(m, n) = 1$. Since $m^{r\phi(n)+1} \equiv m \pmod{n}$ it is enough to consider the case where $b = p^l$.

Let $x = b^{r\phi(n)+1}$. To show that $x \equiv b \pmod{n}$, it is enough to show

$$x \equiv b \pmod{p} \text{ and } x \equiv b \pmod{q}.$$

This is obviously true mod p, since $b \equiv 0 \pmod{p}$. Since $\phi(n) = (p-1)(q-1)$

$$x = (p^l)^{r\phi(n)+1} = \left((p^l)^{q-1}\right)^{r(p-1)} (p^l) \equiv p^l \pmod{q}$$

by Euler's Theorem.

For example let's solve $x^{113} = 341 \pmod{1105}$.
We can factor $1105 = 5(13)(17)$. Thus $\phi(1105) = 4(12)(16) = 768$.

$$\begin{aligned}
768 &= 6(113) + 90 \\
113 &= 90 + 23 \\
90 &= 3(23) + 21 \\
23 &= 21 + 2 \\
21 &= 10(2) + 1
\end{aligned}$$

$$\begin{aligned}
1 &= 21 - 10(2) \\
&= 21 - 10(23 - 21) \\
&= 11(21) - 10(23) \\
&= 11(90 - 3(23)) - 10(23) \\
&= 11(90) - 43(23) \\
&= 11(90) - 43(113 - 90) \\
&= 54(90) - 43(113) \\
&= 54(768 - 6(113)) - 43(113) \\
&= 54(768) - 367(113)
\end{aligned}$$

Thus $113(-367) - 768(-54) = 1$. The general solutions are $s = -367 + 768n$, $t = -54 + 113n$ Letting $n = 1$ we get the nonnegative solution $s = 401$, $t = 59$. Thus 341^{401} is a solution mod 1105.

Calculate $341^2 \equiv 256 \pmod{1105}$
$341^4 \equiv 341 \pmod{1105}$
$341^8 \equiv 256 \pmod{1105}$
$341^{16} \equiv 341 \pmod{1105}$
$341^{32} \equiv 256 \pmod{1105}$
$341^{64} \equiv 341 \pmod{1105}$
$341^{128} \equiv 256 \pmod{1105}$
$341^{256} \equiv 341 \pmod{1105}$
Since 401=256+125+16+4,

$$341^{401} \equiv (341)(256)(341)(341) \equiv 256 \pmod{1105}$$

Thus 256 is the desired solution.

Warning–COMPUTING $\phi(n)$ IS HARD

Although this seems like an easy algorithm, there is one serious difficulty. We need to compute $\phi(n)$. If we can factorize n, then this is easy. But if n is very large, factorizing n might be very hard and computing $\phi(n)$ might be just as hard.

Suppose p and q are primes and $n = pq$. We claim that if we know n and $\phi(n)$ then we can easily computer p and q. Note that that

$$\phi(n) = \phi(pq) = (p-1)(q-1) = n - p - q + 1.$$

Thus we can compute

$$p + q = n + 1 - \phi(n).$$

Next note that p and q are the roots of the quadratic equation

$$X^2 + (p+q)X - n = 0.$$

We can solve this using the quadratic formula

$$p, q = \frac{-(p+q) \pm \sqrt{(p+q)^2 - 4n}}{2}$$

Thus if we know n and $\phi(n)$ we can factor n.

3 Public Key Cryptography

We describe the RSA public key cipher.[1]

Let p and q be large primes and let $n = pq$. Choose m such that $\gcd(m, \phi(n)) = 1$. We publish m and n.

Suppose Alice wants to send us a message. We assume the message is a sequence of numbers $a_1, \ldots, a_k$ where each a_k is an integer less than n. She calculates $b_1, \ldots, b_k$ where $b_i \equiv a_i^m$ and send us $b_1, \ldots, b_k$.

When we receive $b_1, \ldots, b_k$. We use the method of §2 to solve the equations $x_1^m \equiv b_i \pmod{n}, \ldots x_k^m \equiv b_i (\mathrm{mod}\ n)$. In otherwords, we calcluate $s, t \geq 1$ such that $ks - \phi(n)t = 1$ and take $x_i = b_i^s$.

We claim that we must have $x_i \equiv a_i \pmod{n}$. By Theorem 2.1 the function $x \mapsto x^m \pmod{n}$ is an onto map from $\mathbb{Z}_n$ to $\mathbb{Z}_n$. It follows that it is also onto (if $x \neq y$ and $x^m = y^m$ we would have to miss something else in the image. Thus we have found a_i.

What if someone intercepted Alice's message. Without knowing $\phi(n)$, they would have no idea how to find the s that we use to decode the message. As we argued above, fining $\phi(n)$ from n is equivalent to being able to factor n, and there is no known efficient way to do this.

Example 1 Suppose we take each of the 26 letters of the alphabet and represent them as a two digit string. We could take A=01, B=02,...,Z=26. So DOG becomes the string 041507.

[1] This is also described in §5.3 of Jones & Jones.

Let $p = 5$ and $q = 11$ then $n = 55$ and $\phi(n) = 40$. Since $\gcd(3, 40) = 1$ we could take $k = 3$. We could take the message $a_1 = 4, a_2 = 15, a_3 = 7$. And code it as $b_1 = 9, b_2 = 20, b_3 = 13$.

To decode the message we need to know how to solve the equation $3s - 40t = 1$. One solution is $(27, 2)$. Thus we can decode the message by $a_i = b_i^{27}$ $(mod 55)$. We get back 4, 15, 7.

In practice we will have p and q large (over 100 digits), so we can code a block of letters as each a_i. Here is another exampe

Example 2 Take $p = 757$ and $q = 541$. Then $n = 409537$ and $\phi(n) = 408240$ Let $k = 1327$. Then $\gcd(k, \phi(n)) = 1$. We can send our message DOG with a single $a = 4157$. Then

$$b \equiv (4157)^{1327} \equiv 275196 \pmod{409537}$$

To decode messages we need to solve $1327s - (408240)t = 1$. We can do this using the Euclidean Algorithm. One solution is $s = 28303$, $t = 92$.

Thus we can decode our message by taking $a = b^{28303}$.