Number Theory Section Summary: 7.5
The RSA algorithm

The RSA algorithm (developed by Rivest, Shamir, and Adleman, in 1977) depends on the fact that

Computers can't quickly and efficiently factor humongous numbers.

So here are the steps:

- Come up with two large primes: p and q. Don't tell a soul your secret numbers!
- Your public modulus will be $n = pq$.
- Come up with a value k such that $\gcd(k, \phi(n)) = 1$ (easy choice: a prime bigger than either p or q: since $\phi(n) = (p-1)(q-1)$, k must be relatively prime to $\phi(n)$; unfortunately, it would be horrendously large!).
- Publish (k, n) – these are your public keys. You can tell the whole world!
- Compute the *decripting* (or *recovery*) *exponent* j as the solution of

$$kj \equiv 1 (\text{mod } n)$$

  To do this, you can use the result of exercise #8(a), p. 139: if $\gcd(a, n) = 1$, then the linear congruence $ax \equiv b (\text{mod } n)$ has the solution $x \equiv ba^{\phi(n)-1} (\text{mod } n)$.

- If one want to send you an encrypted message, they
  - Write their message as a number, using ASCII or some other coding (such as the one on page 148), and
  - send

$$\boxed{r \equiv M^k (\text{mod } n).}$$

- You decode the message as

$$M \equiv r^j (\text{mod } n).$$

$$r^j \equiv (M^k)^j \equiv M^{kj} \equiv M^{1+\varphi(n)t}$$
$$\equiv M M^{\varphi(n)t} \equiv M \left(M^{\varphi(n)}\right)^t \equiv M \cdot 1^t$$
$$\equiv M \pmod{n}$$