Lecture notes on RSA and the totient function

Jason Holt
BYU Internet Security Research Lab*

8 October 2002

RSA takes advantage of Euler's generalization of Fermat's Little Theorem, namely:

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

1 Euler's Totient Function

Euler's totient function, $\phi(n)$ is defined as follows, where $p_0..p_k$ are the prime factors of n. Given

$$n = p_0^{e_0} \cdot p_1^{e_1} \cdot \ldots \cdot p_k^{e_k},$$

$$\phi(n) = (p_0 - 1)p_0^{e_0-1} \cdot (p_1 - 1)p_1^{e_1-1} \cdot \ldots \cdot (p_k - 1)p_k^{e_k-1}.$$

For example:

$$90720 = 2^5 \cdot 3^4 \cdot 5 \cdot 7$$

$$\phi(90720) = (2-1)2^4 \cdot (3-1)3^3 \cdot (5-1)5^0 \cdot (7-1)7^0 = 20736.$$

The totient function describes the number of values less than n which are relatively prime to n. For the purposes of RSA, we're only concerned with values of n which are the product of 2 primes, p and q, so $\phi(n)$ is always just $(p-1)(q-1)$.

2 Encryption and decryption with RSA

Encryption and decryption in RSA take advantage of the fact that for a message m and exponents e and d:

$$m^{ed} \equiv m \pmod{n}$$

This works because e and d are chosen such that for some (unimportant) value k,

$$ed = k\phi(n) + 1$$

(That is to say, $ed \equiv 1 \pmod{\phi(n)}$.) Since any

$$m^{k\phi(n)} = m^{\phi(n)} \cdot m^{\phi(n)} \cdot \ldots \cdot m^{\phi(n)} = 1 \pmod{n},$$

$$m^{ed} = m^{k\phi(n)+1} = m \cdot m^{k\phi(n)} = 1 \cdot m \pmod{n}$$

Since ed is congruent to $1 \bmod \phi(n)$, d happens to be the multiplicative inverse of $e \bmod \phi(n)$ (and vice versa). e is chosen somewhat arbitrarily, usually as something inexpensive when used as an exponent. Nowadays, it's generally 65537. e is known as the public or encryption exponent, and d as the decryption or private exponent. To encrypt a message m to a ciphertext c, simply calculate

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

Since only the recipient knows d, only he can recover the message:

$$m = c^d = (m^e)^d = m^{ed} \pmod{n}$$

Why is RSA secure? Well, because both discrete logarithms and factoring are hard for large numbers. Given

$$m^e \pmod{n}$$

it's hard to determine what either the base or exponent were (although in the case of RSA, the public exponent is published). And given the public modulus n and public exponent e, it's hard to compute d because you can't calculate $\phi(n)$ without knowing n's factors p and q.

3 Signing with RSA

RSA can be used to produce digital signatures on the hash h of a message m. The signer raises h to his secret exponent d:

$$s = h^d \pmod{n}.$$

Only she knows how to do this because only she knows d. Anyone else can verify the signature by raising it to the public exponent:

$$h = s^e = (h^d)^e \pmod{n}.$$