Appendix E

RSA Asymmetric Encryption

Of all encryption algorithms, the RSA asymmetric algorithm is the easiest to understand—for the mathematically inclined. It was invented by Rivest, Shamir, and Adleman in 1978 [RSA78], and the following development is based in part on [Sch90]. This appendix presumes knowledge of basic number theory.

Euler's Theorem

RSA is based on Euler's theorem from number theory. Recall that a prime number has no factors other than itself and unity, and two numbers are relatively prime if they have no common factors other than unity. For an integer k, define the *totient function* $\phi(k)$ as the number of positive integers that are less then k and are relatively prime to k, including unity.

Example... The prime factors of 10 are 2 and 5. Thus the integers less than 10 that are relatively prime to 10 are 1,3,7, and 9, and thus $\phi(10) = 4$.

Euler's theorem states that if n and m are relatively prime, then $m^{\phi(n)} \bmod n = 1$.

Example... 10 and 7 are relatively prime, and

$$7^{\phi(10)} \bmod 10 = 7^4 \bmod 10 = 2401 \bmod 10 = 1 .$$

Finding the Keys

An asymmetric encryption algorithm needs two keys, one of which is kept secret and the other can be made public. In the RSA algorithm, these keys are obtained by construction, starting with two large prime numbers p and q. The security of the algorithm depends on the assumption that, although their product $n = p \bullet q$ is easy to calculate, the factorization of n is computationally infeasible if p and q are large.

With p, q and n in hand, the two keys can be constructed. First, it can be shown that $\phi(n) = (p-1) \bullet (q-1)$. Choose any integer s that is relatively prime to $\phi(n)$. The public key is then (n, s).

To find the secret key, let t satisfy $s \bullet t \bmod \phi(n) = 1$; that is, t is the multiplicative inverse of s, modulo $\phi(n)$. In fact, t can be obtained directly from the public key by

$$t = s^{\phi(\phi(n)) - 1} \bmod \phi(n),$$

since then, by Euler's theorem,

$$t \bullet s = s^{\phi(\phi(n))} \bmod \phi(n) = 1 .$$

The secret key is now (n, t). In principle this secret key can be obtained from the public key. However, this requires knowing $\phi(n)$, which in turn requires the prime factors of n. By assumption, it is computationally infeasible to determine the factors of n.

Encryption and Decryption Algorithms

A plaintext P is presumed to be a non-negative integer less than n, so that $P \bmod n = P$. The ciphertext (encrypted version of P) is another integer C determined by the encryption algorithm (requiring the public key)

$$C = P^s \bmod \mathrm{n}$$

and the plaintext can be recovered by the decryption algorithm (requiring the secret key)

$$P = C^t \bmod n.$$

To verify that decryption is the inverse of encryption,

$$C^t \bmod n = P^{st} \bmod n = P^{\phi(n) \bullet k + 1} \bmod n \text{ for some } k,$$

and by Euler (presuming that P and n are relatively prime)

$$P^{\phi(n)} \bmod n = 1,$$

and thus

$$C^t \bmod n = P \bmod n = P.$$

Complications

P and n must be relatively prime, and thus there are “only” $\phi(n) = (p-1) \bullet (q-1)$ acceptable plaintext’s. However, the chance of choosing a forbidden plaintext is very small, since only $p + q - 1$ plaintexts out of $p \bullet q$ are forbidden—a diminishing fraction as n gets large. Thus, this problem is safely ignored.

The requirement that s and $\phi(n)$ be relatively prime is more troublesome, since it requires a factorization of $\phi(n)$, and hence $(p-1)$ and $(q-1)$, be available when the public key is chosen. Fortunately, there are techniques for insuring that these factors are available by construction.

Actually computing the encryption and decryption algorithms is tricky because the numbers involved are gigantic. The algorithm must be decomposed into smaller pieces using the properties of the modulo function, such as

$$i \bullet k \bmod \mathrm{n} = (i \bmod n) \bullet (k \bmod n) \bmod n,$$

$$P^{i \bullet k} \bmod n = (P^i \bmod n)^k \bmod n.$$

All the steps can be decomposed into the modulo product or sum of two integers, which is more easily handled (see exercises).

Example... Even for the simple case $n = 5 \cdot 11 = 55$ the computation becomes tricky on a standard spreadsheet due to overflow problems. For this case, $\phi(55) = 4 \cdot 10 = 40$ and only 40 out of the 55 possible messages are acceptable (because n is small). An acceptable public key is

(55,13), because 13 is relatively prime to 40, and the corresponding private key is (55,37). (You can verify that $(13 \cdot 37) \bmod 40 = 1$.) Now the encryption/decryption algorithm is

$$C = P^{13} = P^{1+4+8} = P \cdot P^4 \cdot P^8$$

$$P = C^{37} = C \cdot C^4 \cdot C^{32}$$

where the (mod 55) reduction is presumed everywhere. Now the encryption algorithm can be implemented by calculating P^2, P^4, and P^8 successively, each term calculated by squaring the last and reducing the result (mod 55). Finally, taking the product (iteratively in pairs if necessary) and reducing the result (mod 55) yields C. A similar algorithm performs the decryption. Using this approach, the largest number encountered is 54^2, and arithmetic overflow is not an issue.

Blind Signature Algorithm

A blind signature algorithm suitable for anonymous digital cash as described in "Privacy and Digital Cash" (Section 14.4.2 on page 379) can be based on RSA as follows. The consumer wants to get the issuer to encrypt a message digest MD of a digital cash token with the issuer's secret key (n, t), but without the issuer seeing MD. The consumer knows the issuer's public key (n, s) and chooses a random integer $0 < B < n$, called the blinding factor. Instead of providing MD to the issuer, he provides $B^s \cdot MD \bmod n$. The issuer than encrypts this with its secret key,

$$((B^s \cdot MD) \bmod n)^t \bmod n = (B \cdot MD^t) \bmod n \qquad \text{EQ 33}$$

and return result to the consumer. The consumer can simply divide by B (or more properly multiply by the $\bmod n$ multiplicative inverse of B) which he knows but the issuer doesn't, to recover the signature $MD^t \bmod n$.

When the issuer later receives its digital cash token and signature, it can check its own signature to be sure the token is genuine. The issuer can also check the token identifier to be sure this is the first spending. However, since this is the first time the issuer has seen MD, it can't crossmatch MD's to violate the privacy objective.

Exercises

EE.1. For the RSA encryption algorithm, let the public key be (7,33).

a. Find the secret key.

b. What is the range of possible messages M?

c. For $E = 20$, what is M?

EE.2. For the RSA encryption algorithm, let the public key be (151,323). What is the message M corresponding to $E = 215$?

Hint: It is recommended that you use a spreadsheet program. It provides a `mod(k,n)` function that is handy. The variables in your spreadsheet program will overflow if you follow a straightforward path. It is recommended that you decompose the exponents into binary numbers and then break the problem up into the smallest pieces you can.

EE.3. Show that $\phi(p \bullet q) = (p-1) \bullet (q-1)$ when p and q are prime..

EE.4. Verify Euler's theorem for $n = 16, M = 3$.

EE.5. Prove the validity of the RSA blind signature algorithm.