

## RSA Crypto-system

In this note we measure the computing time by the number of digits,  $k$ , to express the given number  $x$ . We refer to the value of  $k$  as the size and the value of  $x$  as the magnitude. If the time is proportional to a polynomial of  $k$ , it is called polynomial time. If it is proportional to  $x$  it is called exponential time.

**Setting up:** Prepare large prime numbers  $p$  and  $q$ , 100 digit each. To generate such big prime numbers, use Solovay and Strassen's probabilistic primality test. This test can be done in polynomial time. Compute  $n = pq$ . This multiplication can be done in polynomial time. An attacker will face exponential time for factoring.

The Euler function  $\phi(n) = (p-1)(q-1)$  can be computed in polynomial time. Then generate a large random number  $d$  for decryption, which should be kept secretly. Compute  $e$  such that  $ed \equiv 1 \pmod{\phi(n)}$ . That is,  $e$  is the multiplicative inverse of  $d$  modulo  $\phi(n)$ . This part is done by the Euclidean algorithm in polynomial time.

**Public key (e, n)**

**Secret key (d, n)**

This setting-up is done for every subscriber. We specify subscribers by ( ) for the above keys.

### Communication from A to B

**Encryption:** Suppose the plain message is encoded into a large number  $M$  between 0 and  $n(B)-1$ . Using B's secret key ( $e(B), n(B)$ ), A computes the following cryptogram  $C$ .

$$C = M^{e(B)} \pmod{n(B)}$$

**Decryption:** B decrypts  $C$  using his secret key as follows:

$$D = C^{d(B)} \pmod{n(B)}$$

Encryption and decryption can be done using repeated squaring and division in polynomial time. Actually squaring and division for mod must be done alternately, so the intermediate values will not exceed the limit.

**Theorem.**  $D = M$ . After decryption, we have the original message. To prove this, we use Fermat's theorem.

**Authentication:** If A wants B to trust him/her, A sends the following.

$$C = (M^{d(A)} \pmod{n(A)})^{e(B)} \pmod{n(B)}$$

The receiver B computes PROOF first, using B's secret key.

$$\text{PROOF} = C^{d(B)} \pmod{n(B)}$$

Then B computes  $M = \text{PROOF}^{e(A)} \pmod{n(A)}$ , using A's public key.