Multi–precision Arithmetic and Data Encryption

A primary use of multi-precision arithmetic is for *data encryption*. A *plaintext* message or data file M is modified by some well-known mathematical algorithm to render the message unreadable and non-sensical. The encrypted message or file is called the *ciphertext* message and is denoted by C. Clearly, there must be some way for the original message M to be reconstructed from the ciphertext message (a processed called *decryption*), and equally clearly this reconstruction must be difficult or impossible for all but the intended recipient of the message.

There are two fundamentally different approaches to encryption and decryption.

1. Symmetric Key encryption uses a single mathematical constant (the key) to perform the encryption algorithm, and the same constant to perform the inverse encryption (decryption). Clearly, when this approach is used both the sender and receiver of an encrypted message must have some way to securely exchange the key. There are a number of examples of symmetric key encryption including *Data Encryption Standard* (*DES*) and the *Advanced Encryption Standard* (*AES*) (also known as *Rijndael*), and many others.

2. Public Key encryption uses two different keys, called the *public* key and the *private* key. The two keys are generated by a mathematical algorithm (described below), and then the public key is placed in a well-known and publicly accessible repository (called the *Key Repository*). The private key is of course kept private and not disclosed. Anyone wanting to send an encrypted message to a recipient simply uses that recipient's universally known public key to execute the public key encryption algorithm. After being encrypted with the public key, the message is undecipherable by anyone excepting the owner of the corresponding private key. The most famous example of this type of encryption is called the *Rivest, Shamir and Adleman* (*RSA*) algorithm designed by the three authors and published in 1978. The *RSA* algorithm is described in detail below.

The RSA Public-Key Encryption Algorithm The use of the RSA algorithm requires three distinct steps, the key generation step, the encryption step, and the decryption step.

1. Choose the public and private keys as follows:
   (a) Choose two random and distinct prime numbers p and q. The size (number of bits) for p and q should be about the same.
   (b) Compute $n = pq$
   (c) Compute $\phi(n) = (p-1)(q-1)$
   (d) Choose an integer d such that $1 < d < \phi(n)$ and $\texttt{gcd}(d, \phi(n)) = 1$.
   (e) At this point, the pair n and d are the public key and can be freely shared with anyone.
   (f) Compute e as the *multiplicative inverse* of $d \; mod(\phi(n))$
   (g) At this point, the pair n and e are the private key, and should not be divulged.
2. Perform the actual encryption as follows:
   (a) Convert the plaintext message into an integer m such that $0 < m < n$. The simplest way to do this is to strip the first $k-1$ bits from M, where k is the size in bits of n. If the plaintext message is much larger than n, simply perform the encryption a number of times of successive m values from the original plaintext message M.
   (b) Compute the ciphertext message $c = m^d (mod\ n)$. Remember that n and d are the public key.
3. To decrypt the message c, simply compute $m = c^e (mod\ n)$.

The security of the RSA algorithm depends on the relative difficulty of factoring n into p and q. If the attacker can compute these, the he can easily compute the private key pair n and e.