Public Key Cryptography

Public Key cryptography: The basic idea is to do away with the necessity of a secure key exchange, which is necessary for all private key encryption schemes. The idea is as follows:

1) Bob creates two keys, a public key, E and a private key D.

2) Bob posts the public key in a location that anyone can access.
The important thing here is that the knowledge of E does not compromise the value of D in any way shape or form.

3) Now, anyone who wants to send a message to Bob encrypts it using the public key E.

4) Bob can now read the message using his private key D. However, since the value of E gives no information as to the value to D, all others can not read the message.

The idea seems easy enough, but the difficulty is in finding some mathematical function to use in this scheme such that the knowledge of E does not compromise the secrecy of D. Clearly in all the other schemes we have seen, knowledge of the encrypting key all but completely gives away the decrypting key.

One thing to note however is that if you use a system outlined above with nothing extra, although Bob can decipher a message sent to him, he can not be sure of who the sender is, because the whole public has the ability to encipher a message, so someone could easily indicate in their message that they were someone else and Bob would not have any way of knowing. But, the person sending the message can be confident that no one read can read the plaintext except for Bob, the only person with the private key.

RSA Cryptosystem

Now, we are ready to look at the RSA algorithm:

1) Pick large primes p and q.
2) Compute $n=pq$ and $\phi(n)=(p-1)(q-1)$
3) Pick a value e such that $\gcd(e, \phi(n)) = 1$. (Note that this is fairly easy to do by randomly picking values e and testing them with Euclid's algorithm until you find one that works.)
4) Compute d such that $ed \equiv 1 \pmod{\phi(n)}$. You are guaranteed to be able to do this by the extended Euclidean algorithm.
5) Public keys : e, n
Private key : d, (n is also necessary for decryption, but is clearly public...)

Encryption function : $E_{n,e}(x) = x^e \pmod{n}$
Decryption function : $D_{n,d}(y) = y^d \pmod{n}$

Now, we must verify that this is a valid encryption scheme:

$D_{n,d}(E_{n,e}(x)) = D_{n,d}(x^e) = (x^e)^d \pmod n = x^{ed} \pmod n$

Now we will invoke the given information about the product ed:

$ed \equiv 1 \pmod{\phi(n)}$, thus, we can find an integer k such that $ed = k\phi(n) + 1$.

Now we have the following:

$x^{ed} \pmod n = x^{k\phi(n)+1} \pmod n$
$= x^{k\phi(n)}x^1 \pmod n$
$= (x^{\phi(n)})^k x \pmod n$
$= (1)^k x \pmod n$, invoking Euler's formula.
$= x \pmod n$, proving that the encryption scheme is valid, assuming that the gcd(x,n) = 1.

Since you are picking large primes the probability that gcd(x,n) =1 is quite high. (If you pick 20 digit primes for both p and q, the probability is roughly $1 - 10^{-20}$ that gcd(x,n) = 1.

In practice, one would have to have a reliable way to convert the data they wanted to send (maybe a bitstring or some text) into a number less than n. An example of such a scheme would be treating a block of letters (assume all lower case) as a number in base 26. For example, "break" would equal $1x26^4 + 17x26^3 + 4x26^2 + 0x26^1 + 10x26^0$, using 'a' = 0, 'b' = 1, …, 'z' = 25. It would be natural to make the valid block size, k, where 26^k is less than n but 26^{k+1} is greater than n. Many other schemes would work as well. But typically, the ciphertext is represented as integers instead of being converted to letters.