The RSA Public Key Encryption System

Common Sense tells us that if we know how a secret message was encoded then we can easily decode it. As an example suppose we are given the message

DPODSFUF NBUIFNBUJDT

and are told that it was encoded by replacing each letter in the original message by the one that immediately follows it in the alphabet. To decode the message all we have to do is replace each letter by the one that preceeds it yielding

CONCRETE MATHEMATICS.

For most of history this fact (the equivalence of coding and decoding) meant that the security of a coding system revolved around keeping the encryption process secret. This usually led to one of two situations: The first was that only highly trusted individuals were allowed access to the encoding key. This meant that the code could not be used by many people. The second was just the opposite, that many people were given the secret. The problem with this is that it effectively destroyed the code's ability to be used for sending highly secure information.

A further problem is that we can think of many situations where we would like the entire world to be able to use the code to send information without anyone (but the intended receiver) being able to read it. A good example would be a brokerage house with many large clients, all of whom want to keep their dealings secret. Obviously, if our view of the situation was correct, the brokerage house would have to create a different code for each of their clients, an obviously messy affair. Fortunately, in this case common sense led us astray. There are algorithms known as Public Key Cryptosystems for which knowing the encryption key does not give much information about the decryption one. The reason that this type of algorithm is known as a Public Key Cryptosystem is that we can make the encryption key public so that the entire world can use it while still preserving the security of the code. In the example we just gave, when a customer wants to send a buy order all she has to do is look up the brokerage's encoding key in the Public Code Directory (The Black Pages?) and use it to send the message. When the brokerage receives the message they use the decoding key, which only they know, to translate it back into plaintext. The remainder of these notes will be devoted to describing the RSA Public Key Cryptosystem [**2**].

The RSA scheme works as follows: We choose two large primes p, q and set $N = p \cdot q$. Recall that $\phi(n) = (p-1)(q-1)$ is the number of integers less than N that are relatively prime to N. Choose positive integers $e, d < \phi(N)$ such that

$$\gcd(e, \phi(N)) = 1, \qquad e \cdot d \equiv 1 \pmod{\phi(n)}.$$

The numbers e, N are made public, d is kept private. The message to be encoded is some integer X less than N (if the message is large, we split it up into many blocks of size $< N$ and encode each of them separately). The encoding and decoding functions are given by

$$E(X) = X^e \bmod N, \qquad D(Y) = Y^d \bmod N.$$

The sender sends the encoded message $Y = E(X)$ to the receiver who decodes it by calculating $D(Y)$.

There are three major issues involved in showing that this encoding scheme is useful. First, we have to show that it is practical: We can find p, q, e, d possessing the properties given above. Second, we have to show that it is secure: It will be difficult for anyone who doesn't know d to decode the message. Finally, and most importantly, we have to show that it is correct: $D(E(X)) = X$.

(i) *Practicality:* There are (probabilistic) fast algorithms for finding prime numbers, so we can find large primes p, q. Since $\phi(N) = (p-1)(q-1)$ it has no prime factors greater than $\max(p,q)$. Thus we can choose e as any prime larger than $\max(p,q)$ and have $\gcd(e, \phi(N)) = 1$. We can then use the generalized version of Euclid's algorithm to find d such that $e \cdot d \equiv 1 \pmod{\phi(N)}$.

(ii) *Security:* Given the information available publically there are two ways that this code can be broken by an adversary. Either he can try to factor N into its two components (which along with knowledge of e will enable him to calculate d), or he can try to invert the value $E(X)$, that is to use the fact that $e, N, E(X)$ are known to find the one $X < N$ such that $X^e \equiv E(X) \pmod{N}$. Fortunately both of these problems are so hard that despite many years spent in the attempt mathematicians have not been able to come up with practical algorithms for solving either of them.

(iii) *Correctness:*

Theorem:

$$D(E(X)) = X.$$

Proof:

$$D(E(X)) \equiv (X^e \bmod N)^d \equiv X^{ed} \pmod{N}.$$

Since $e \cdot d \equiv 1 \pmod{N}$ there exists an integer k such that $e \cdot d = k\phi(N) + 1$ and thus

$$D(E(X)) \equiv X^{k\phi(N)+1} \pmod{N}.$$

There are now three possible cases that have to be examined:

(a) $X \perp N$: By Euler's extension to Fermat's little theorem ([**1**], p. 133) $X^{\phi(N)} \equiv 1 \pmod{N}$ so

$$X^{ed} \equiv X^{k\phi(N)+1} \equiv X \pmod{N}.$$

(b) $X = i \cdot p; \quad 0 < i < q$.: Since $(ip) \perp q$ Fermat's little theorem tells us that $(ip)^{q-1} \equiv 1 \pmod{q}$ so

$$X^{ed} \equiv (ip)^{(q-1)(p-1)k+1} \equiv (ip) \equiv X \pmod{q}.$$

Furthermore $X \equiv 0 \pmod{p}$ so

$$X^{ed} \equiv 0 \pmod{p}.$$

Equation 4.42 in [**1**] tells us that if $p \perp q$ then

$$a \equiv b \pmod{pq} \quad \Leftrightarrow \quad a \equiv b \pmod{p} \quad \text{and} \quad a \equiv b \pmod{q}.$$

Thus we see that

$$X^{ed} \equiv X \pmod{N = pq}.$$

which is what we wanted to prove.

(c) $X = i \cdot q; \quad 0 < i < p$.: The proof is similar to (b).

References.

[**1**] Graham, R., Knuth, D., Patashnik, O., *Concrete Mathematics,* Addison Wesley, 1988.

[**2**] Rivest, R., Shamir, A., and Adelman, L, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Communications of the ACM,* **21** pp. 120-126, 1978.