20. Public Key Cryptography and RSA

Erica started out by giving the answer to the homework question from last time:

Example: Decode the message PDQZPSFA using the decryption function \( \delta(x) = 21x + 3 \mod 26 \).

The answer is **GO BIG RED**. For example, the letter P corresponds to the number 15, and 15 decodes as \( \epsilon(15) = (21)(15) + 3 \mod 26 = 6 \), and 6 corresponds to the letter G. One treats the rest of the message similarly.

We then returned to our discussion of ciphers. Although there are many more affine ciphers than shift ciphers (how many more?), they are really not that much more secure. Eve has only to make two educated guesses (instead of one) and he can recover the value of \( a \) and \( b \) in the encryption function \( \epsilon(m) \). Once he has the encryption function, he can calculate the decryption function just as we did in the examples above. Both shift ciphers and affine ciphers are examples of substitution ciphers, where the ciphertext alphabet is just some (possibly random) permutation of the cleartext alphabet. A serious drawback to substitution ciphers is that one can use common knowledge about the English language (for example, the fact that “e” is by far the most common letter) to make guesses about what the various letters stand for. The “CRYPTOQUOTES” puzzles you find in the newspaper are examples of substitution ciphers.

One might consider encoding more than one letter at a time. For example, suppose we wish to encode 4 letters as one “block”. We can still translate each letter into a two-digit number as above, but then we consider the four letters together as an eight-digit number \( m \). Such a number is certainly less than \( 10^8 \), so we can use \( 10^8 \) as our modulus and proceed as before with our affine cipher. Choose parameters \( a \) and \( b \) with \( \gcd(a, 10^8) = 1 \), and set \( \epsilon(m) = (am + b) \mod 10^8 \). The frequencies for blocks of four letters are not nearly so well known as they are for individual letters, and so this type of affine cipher is much more secure.

There are still problems, however. The main problem is that before Alice and Bob can communicate using this affine cipher, they must decide on the values of \( a \) and \( b \). (These values are called the “key” for the cipher.) They can’t just send these values to each other unencrypted, because then Eve could read them and he would know the formula for \( \epsilon \). So how do Alice and Bob decide on their key?

One solution is to use public key cryptography. The basic idea of a public key system is that even if Oscar knows \( \epsilon \), he can’t figure out \( \delta \). In the RSA cyptosystem, which is the public key system we will focus on, the encryption function has the form

\[
\epsilon(x) = x^e \mod N
\]

where \( e \) and \( N \) are carefully chosen positive integers. It turns out that the decrypting function has the same form: \( \delta(x) = x^d \mod N \). In general, given such an \( e \) it is very difficult to find \( \delta \) in any reasonable amount of time, even with the world’s fastest computers. However, Alice chooses \( N \) in such a way that she (and only she) can quickly compute \( \delta \). The secret lies in the prime factorization of \( N \). Here is how it works:

**Pick primes:** The first step for Alice is to pick two prime numbers \( p \) and \( q \) with \( p \neq q \). In practice, these primes need to be very large — about 150 digits each— for the cipher to be secure.

- We’ll take \( p = 7919 \) and \( q = 7937 \).
Calculate \( n \) and \( k \):: Next Alice simply sets \( N := pq \) and \( k = (p - 1)(q - 1) \). Since Alice knows the factorization of \( n \), she can compute \( k \) easily. Notice that in practice, \( n \) will have at least 300 digits. This means that computing \( k \) would be very difficult without knowing the factorization of \( N \), and factoring \( N \) would also be very difficult.

- In our example, we have \( N = (7919)(7937) = 62853103 \) and \( k = (7918)(7936) = 62837248 \).

Choose \( e \) — the encoding exponent:: The next step is to pick a value of \( e \) at random, making sure that \( \gcd(e, k) = 1 \). Alice does this by first selecting a value for \( e \) and then performing the Euclidean Algorithm to calculate \( \gcd(e, k) \). If this \( \gcd \) is 1, great. Otherwise, Alice simply chooses a new value of \( e \).

- For our example, we’ll just take \( e = q = 7937 \). (We wouldn’t want to do this in practice, of course, as it may give away the factorization of \( N \)!)  

Find \( d \) — the decoding exponent:: Now, Alice needs to find a value of \( d \) so that \( de \equiv 1 \pmod{k} \). She can do this through using the Euclidean Algorithm and back substitution. This could be done by hand for small primes such as mine, but in practice we would do this on a computer.

- Using Maple, I found \( d \equiv e^{-1} \equiv 49607937 \pmod{k} \).

When all is said and done, Alice’s public keys are \( N \) and \( e \), and her private key is \( d \). The encoding function, which anyone in the world can use to send a message to Alice, is \( \epsilon(x) = x^e \pmod{N} \). The decoding function, which only Alice knows, is \( \delta(x) = x^d \pmod{N} \). Since in our example \( N \) is eight digits long and we want our messages to be less than \( N \) (so they will be remainders modulo \( N \)), we should break our message into four letter blocks and encode each of these blocks separately. For example, if we wanted to encode the word MATH, we plug \( m = 12001907 \) into our encryption function to get

\[
\epsilon(12001907) = 12001907^{7937} \pmod{62858103} = 53218748.
\]

One can also check (again, using a computer) that

\[
\delta(53218748) = (53218748)^{49607937} \pmod{62858103} = 12001907.
\]

So we see that the decryption function works (for this message, anyway).

Let’s try another example:

Example: Suppose \( p = 53 \) and \( q = 71 \). Find the encryption and decryption functions, with \( e = q \). Encode and decode the two-letter word BO.

Here, \( N = (53)(71) = 3763 \). Since \( e = q \), the encryption function is:

\[
\epsilon(x) = x^{71} \pmod{3763}.
\]

Now, BO corresponds to the four digit number 0114. Gabe did the grunge work and found that \( \epsilon(114) = (114)^{71} \pmod{3763} = 1463 \). To find the decryption function, we need to find the inverse of \( e \) modulo \( k \), where \( k = (p - 1)(q - 1) = (52)(70) = 3640 \). Mike did the work here (using the Euclidean Algorithm and back substitution) and found that

\[
2871 \equiv 71^{-1} \pmod{3640}.
\]

Thus, the decryption function is

\[
\delta(x) = x^{2871} \pmod{3763}.
\]
To check that $\delta(x)$ does indeed correctly decrypt $\epsilon(114) = 1463$, once again Gabe did the grinding (with checks from Lisa and Huy) to show that

$$\delta(1463) = (1463)^{2871} \mod 3763 = 114.$$ 

So $\delta$ does appear to be the inverse function for $\epsilon(x)$. (The chances that it is not and we just got lucky here are very small indeed!). So why does this work? Next time, we'll prove that that $\delta(\epsilon(x)) = x$ for all $x$.

Here is your homework for Tuesday:

**Homework:** Suppose $\epsilon(x) = x^{4007} \mod 6319$ is the encryption function. Find the decryption function $\delta(x)$. 