Chapter 16

Secret Codes

Since time immemorial, people have found it necessary to send information to others in a secure way, so that even if a message is intercepted by a third party, it can only be read by the intended recipient. This is usually achieved by converting the message into a string of numbers or letters according to some rule that only the sender and recipient know. The process of converting the message into such a string is called "encoding," and the process by which the recipient converts the received string back to the original message is called "decoding."

For example, one of the simplest imaginable encoding rules would be to substitute 01 for an A, 02 for a B, 03 for a C, and so on until we get to 26 for a Z. Then the message

SMALLBRAIN IS THE CULPRIT

gets encoded as the string of numbers

$$1913011212021801091409192008050321121616180920. \qquad (16.1)$$

However, anyone with a modicum of intelligence would be able to break this code in a matter of minutes. In view of the millions of Internet transactions, sensitive business and government communications, and so on, that take place every day, some rather more sophisticated ideas are required to ensure the secure transfer of information in today's world. The codes now in use for such purposes — so-called *RSA codes* — are based on some of the theory of prime numbers and congruence developed in the last few chapters. These codes can be explained in just a few pages, which I shall now attempt to do.

Before going into the theory, let me explain the crucial fact that gives these codes their security. Any form of security is based on things that "attackers" are unable to do. In this case, information is being transmitted electronically by computer, so security needs to be based on something that computers can't do quickly.

Let me now explain something that computers can't do quickly. If you take two reasonably large prime numbers, say 1301 and 2089, you can find their

product (2717789) using a calculator, in a few seconds. However, if instead you were given the number 2717789 and asked to find its prime factors, it would take you and your calculator much longer. (If you doubt this, try using your calculator to find the prime factors of 127741.) The same kind of observation applies if we bring very powerful computers into play: take two very large primes p and q, say with about 150 digits each — I outlined to you how such primes are found in the last section of Chapter 15 — and calculate pq. Keep p and q secret, but tell the most powerful computer in the world today the value of pq. Then however cleverly that computer is programmed (at least with ideas available today), it will probably take at least several centuries for it to find out what p and q are. In other words, computers find factorization of large numbers "difficult."

RSA codes: encoding

Here are the encoding and decoding processes for RSA codes. We choose two large prime numbers p and q, and multiply them together to get $N = pq$. We also find $(p-1)(q-1)$, and choose a large number e, which is coprime to $(p-1)(q-1)$. We then make public the numbers N and e, but keep the values of p and q secret. We shall illustrate the discussion with the values

$$p = 37,\ q = 61,\ N = pq = 2257,\ (p-1)(q-1) = 2160,\ e = 11.$$

(In practice, to ensure the security of the code one uses primes with around 150 digits, but these primes will serve for illustrative purposes.)

The pair (N,e) is called the *public key* of the code. Anyone who wants to send us a message can use the numbers N and e in the following way. First, they convert their message into a string of numbers by some process such as the one given above (01 for A, etc.). They then break up this string into a sequence of numbers with fewer digits than N. For example, with p,q as above, the message (16.1) would be broken up as the sequence

191, 301, 121, 202, 180, 109, 140, 919, 200, 805, 032, 112, 161, 809, 20.

So the message now is a sequence of numbers; call it $m_1, m_2, \ldots, m_k$. The next step is to encode this message. To do this, the sender calculates, for $1 \leq i \leq k$, the value of n_i, where $n_i \equiv m_i^e \bmod N$ and $0 \leq n_i < N$. This can be done quickly on a computer using the method of successive squares described in Example 14.3. The encoded message is the new list of numbers $n_1, n_2, \ldots, n_k$, and this is the message that is sent to us. In the example above, $m_1 = 191$, $N = 2257$ and $e = 11$, and using the method of successive squares, we have

$$191^2 \equiv 369 \bmod 2257,\ 191^4 \equiv 741 \bmod 2257,\ 191^8 \equiv 630 \bmod 2257,$$

and hence

$$191^{11} \equiv 191^{8+2+1} \equiv 630 \cdot 369 \cdot 191 \equiv 2066 \bmod 2257.$$

Hence $n_1 = 2066$. Similarly we find that $n_2 = 483$, $n_3 = 914$, $n_4 = 1808$, and so on. So the encoded message is

$$2066, 483, 914, 1808, \ldots$$

Decoding

Once we have received the message, we have to decode it. In other words, given the received list of numbers $n_1, \ldots, n_k$, we need to find the original numbers $m_1, \ldots, m_k$, where $n_i \equiv m_i^e \bmod N$. But m_i is just the solution of the congruence equation $x^e \equiv n_i \bmod N$, and we have seen in Proposition 15.3 in the previous chapter how to solve such congruence equations: since e has been chosen to be coprime to $(p-1)(q-1)$, we can use the Euclidean algorithm to find a positive integer d such that $de \equiv 1 \bmod (p-1)(q-1)$. Then the solution to the congruence equation $x^e \equiv n_i \bmod N$ is $x \equiv n_i^d \bmod N$. Of course this solution must be the original number m_i, and hence we recover the original message $m_1, \ldots, m_k$, as desired.

In the above example, $e = 11$ and $(p-1)(q-1) = 2160$. Since $n_1 = 2066$, to recover the first number m_1 of the original message we need to solve the congruence equation $x^{11} \equiv 2066 \bmod 2257$. Use of the Euclidean algorithm shows that $1 = 1571 \cdot 11 - 8 \cdot 2160$, so we take $d = 1571$. We then use successive squares to calculate that $2066^{1571} \equiv 191 \bmod 2257$, and hence recover the first number $m_1 = 191$ of the original message. Similarly we recover $m_2, \ldots, m_k$.

To summarise: to encode a message, we raise each of its listed numbers to the power e (e for "encode"), and work out the answer module N; and to decode a received message, we raise each of its listed numbers to the power d (d for "decode"), and work this out modulo N.

Now let's decode a new message (still keeping the above values of p, q and e). You are relaxing on your hotel balcony in Monte Carlo, sipping a pina colada, when your laptop beeps, and the following strange message arrives:

$$763, 28, 1034, 559, 2067, 2028, 798.$$

At once you spring into action, getting your laptop to solve the congruence

equations

$$\begin{aligned}
x^{11} &\equiv 763 \bmod 2257 \quad \to x \equiv 763^{1571} \bmod 2257 \quad \to x \,(\text{mod } 2257) = 251\\
x^{11} &\equiv 28 \bmod 2257 \quad \to x \equiv 28^{1571} \bmod 2257 \quad \to x \,(\text{mod } 2257) = 521\\
x^{11} &\equiv 1034 \bmod 2257 \quad \to x \equiv 1034^{1571} \bmod 2257 \quad \to x \,(\text{mod } 2257) = 180\\
x^{11} &\equiv 559 \bmod 2257 \quad \to x \equiv 559^{1571} \bmod 2257 \quad \to x \,(\text{mod } 2257) = 506\\
x^{11} &\equiv 2067 \bmod 2257 \quad \to x \equiv 2076^{1571} \bmod 2257 \quad \to x \,(\text{mod } 2257) = 091\\
x^{11} &\equiv 2028 \bmod 2257 \quad \to x \equiv 2028^{1571} \bmod 2257 \quad \to x \,(\text{mod } 2257) = 805\\
x^{11} &\equiv 798 \bmod 2257 \quad \to x \equiv 798^{1571} \bmod 2257 \quad \to x \,(\text{mod } 2257) = 04
\end{aligned}$$

This gives the decoded message as the following string of numbers:

$$251, 521, 180, 506, 091, 805, 04.$$

Using the original substitutions 01 for A, 02 for B, etc., you finally translate this into the urgent message:

YOUREFIRED

Unruffled, you saunter back to your drink. You'd been planning to take up that managing directorship offer anyway . . . . .

Security

How secure is the RSA code described above? In other words, if an encoded message is intercepted by a third party (who knows the publicly available values of N and e), how easy is it for them to decode the message? Well, at present the only known way to decode is first to find the value of $(p-1)(q-1)$, and then to calculate d and use the decoding method above, working out d^{th} powers modulo N.

However, $(p-1)(q-1) = pq - p - q + 1 = N - (p+q) + 1$, so if we know N and can find $(p-1)(q-1)$, then we can also find $p+q$. But then we can also find p and q, since they are the roots of the quadratic equation $x^2 - (p+q)x + N = 0$.

In other words, in order to be able to decode messages, a third party needs to be able to find the prime factors p and q of N. But as I explained before starting the description of RSA codes, if p and q are very large primes — both with about 150 digits — then no computer on earth will be able to find p and q, given only their product. In other words, the code is secure!

Even if increases in computer power enable us to factorize products of two 150-digit primes in the future, it will be a simple matter to retain security just by increasing the number of digits in our primes p and q. What is not clear is whether someone will discover a clever new method of computer factorization of large numbers in the future that makes all these codes insecure.