Cryptography and Number Theory!

Shuayb Mohammed

People have been trying to communicate secretly for millennia. In the era of digital communication, it's more important than ever. When you hear cryptography, you might consider the Enigma machine used by Nazi Germany during World War II, with its complex system of rotors and the infamous struggle to crack it. This technique was very secure, but it wouldn't be appropriate for day-to-day use.

The problem is both parties needed to have a copy of the machine, as well as having agreed in secret what settings to start the machine on – i.e. they must have communicated secretly before they can begin using the Enigma to communicate secretly! This presents somewhat of a paradox, and it would hardly be convenient to send an Enigma to Amazon so you can do some shopping. In the 1970s, a new concept revolutionised cryptography: public-key cryptosystems, which enabled secure communication without prior contact in secret.

So, how does it work, broadly speaking? First, we should define some terms.

Plaintext is the information you want to convey.

Ciphertext is the scrambled form of this information.

Encryption turns plaintext into ciphertext.

Decryption reverses encryption.

A *cipher* is an algorithm for encryption and decryption.

Decryption keys and *encryption keys* are pieces of information used in a cipher.

Public-key cryptosystems use a *public* encryption key, which is known to everyone, and a different, but related, *private* decryption key. This means anyone can encrypt a message and send it securely to you, but only you, who possess the private key, can decrypt it and read the plaintext. But, I hear you say, shouldn't it be easy to determine the private key from the public key, if decryption is simply the reverse of encryption? The solution lies in "trapdoor" functions, which are easy to compute, but whose inverses are near impossible to compute, unless you know a particular secret about the function. Now, let's dive into the mathematics underlying one such trapdoor function.

We'll need a lot of number theory. There are some basic ideas in number theory that we'll combine to construct the RSA public-key cipher.

$$\begin{aligned}
\gcd(a,b) = k &\text{ means that } k \text{ is the largest integer dividing } a \text{ and } b\\
\text{if } \gcd(a,b) = 1, &\text{ we say that } a \text{ and } b \text{ are coprime}\\
a \equiv b &\mod n \text{ means that } a = b + kn \text{ for some integer } k\\
\text{if } ax \equiv 1 &\mod n, \text{ we say that } x \text{ is an inverse of } a \text{ modulo } n\\
\text{if } a \equiv b &\mod n, \text{ then } a + c \equiv b + c \mod n\\
\text{if } a \equiv b &\mod n, \text{ then } ac \equiv bc \mod n\\
\text{if } a \equiv b &\mod n, \text{ and } 0 \leq b < n, \ b \text{ is a } \textbf{reduced} \text{ modulo } n
\end{aligned}$$

Euclid's algorithm provides a way of computing $\gcd(a,b)$. If we want to compute $\gcd(700,125)$, we notice that $700 \equiv 75 \mod 125$. Any integer dividing both 700 and 125 must divide 75, so $\gcd(700,125) = \gcd(125,75)$. This allows us to reduce the problem to a simpler form, which we can do repeatedly as shown below to find the solution: 25.

$$\begin{aligned}
700 = 5 \times 125 + 75 &\implies 700 \equiv 75 \mod 125\\
125 = 1 \times 75 + 50 &\implies 125 \equiv 50 \mod 75\\
75 = 1 \times 50 + 25 &\implies 75 \equiv 25 \mod 50\\
50 = 2 \times 25 + 0 &\implies 50 \equiv 0 \mod 25
\end{aligned}$$

By reversing this algorithm we can find an integer solution to $700x + 125y = 25$, as shown below. This is equivalent to solving $700x \equiv 25 \mod 125$. More generally, if $\gcd(a,n) = 1$, we can use this algorithm to solve $ax \equiv 1 \mod n$, i.e. find an inverse of a modulo n.

$$\begin{aligned}
25 &= 75 - 50\\
&= (125 - 50) - (125 - 75)\\
&= (125 - (125 - 75) - (125 - (700 - 5 \times 125))\\
&= (125 - (125 - (700 - 5 \times 125))) - (125 - (700 - 5 \times 125))\\
&= (125 - (-700 + 6 \times 125)) - (-700 + 6 \times 125)\\
&= (700 - 5 \times 125) + 700 - 6 \times 125\\
&= 2 \times 700 - 11 \times 125\\
&\implies x = 2, \ y = -11 \text{ is a solution}
\end{aligned}$$

Next we'll need the φ function. $\varphi(n)$ is defined as the number of positive integers smaller than n which are coprime to it. It's easy to see that, for any prime p, $\varphi(p) = p-1$, as all integers less than a prime cannot share a common factor with that prime. We'll also need to consider $\varphi(m)$ where m is a product of two distinct primes p and q.

q many integers (including m) will be divisible by p

p many integers (including m) will be divisible by q

So, accounting for having counted m twice, there will be $p + q - 1$ many integers that are **not** coprime to m hence

$$\begin{aligned}
\varphi(m) &= pq - q - p + 1\\
&= (p-1)(q-1)\\
&= \varphi(p)\varphi(q)
\end{aligned}$$

Finally, we'll need Euler's Theorem, which states that $x^{\varphi(m)} \equiv 1 \mod m$ if x and m are coprime. This may look daunting, but it's not so hard to prove.
First we consider the set S of integers smaller than m which are coprime to it: $a_1, a_2, \ldots, a_{\varphi(m)}$. Note that we can compute the inverse of any of these elements using Euclid's algorithm. If we pick one of these integers and multiply by x, their product will also share no common factors with x, so it will be an element of S. If we consider two integers in S

$$a_i, \text{ and } a_j$$

and the respective products

$$xa_i, \text{ and } xa_j,$$

then

$$\begin{gathered} xa_i \equiv xa_j \mod m \\ x^{-1}x_i \equiv x^{-1}xa_j \mod m \\ a_i \equiv a_j \mod m \end{gathered}$$

This shows that multiplying by x is *injective*, i.e. multiplying different numbers by x will yield outputs which are different modulo m. Thus multiplying each integer in S simply permutes the set: it takes each element of the set to another element, leaving us with the same set. Unless, of course, $x \equiv 1 \mod n$, in which case it takes each element to itself. As a result, we can use the following reasoning to prove the theorem.

$$\begin{gathered} a_1 \times \cdots \times a_{\varphi(m)} \equiv (xa_1)(xa_2)\ldots(xa_{\varphi(m)}) \\ a_1 \times \cdots \times a_{\varphi(m)} \equiv x^{\varphi(m)}(a_1 \times \cdots \times a_{\varphi(m)}) \\ a_1 \times a_1^{-1} \times \cdots \times a_{\varphi(m)} \times a_{\varphi(m)}^{-1} \equiv x^{\varphi(m)}(a_1 \times a_1^{-1} \times \cdots \times a_{\varphi(m)} \times a_{\varphi(m)}^{-1}) \\ 1 \equiv x^{\varphi(m)} \end{gathered}$$

Now that we have all of these tools, let's take a minute to explore some of the history. The methods of RSA cryptography were first developed covertly in 1973 by Clifford Cocks, working for the British Goverment Communications Headquarters (GCHQ). It was intended as a secure means for military communications, but this development was not declassified until 1997. The concept first arose publicly in 1976, with Whitfield Diffie and Martin Hellman introducing asymmetric cryptography as a new direction for the field.

Inspired by Diffie and Hellman, three professors at MIT (Ron Rivest[1], Adi Shamir[2] and Leonard Adleman[3] got together and dedicated their time to developing a trapdoor function. Rivest and Shamir, computer scientists, were largely responsible for suggesting ideas, of which many failed, while Adleman, the mathematician of the trio, scrutinised their proposals, slowly getting closer to a secure solution. After a year of searching, Rivest had a breakthrough after spending Passover of 1977 drinking at a student's house. The algorithm is named RSA in honour of these three, in the order of their initials on the

[1]Image below is an own work of Ronald Rivest and is licensed under CC BY-SA 4.0
[2]Image below is attributed to the Royal Society and is licensed under CC BY-SA 3.0
[3]Image below is attributed to Leonard Adleman and is licensed under CC-by-SA 3.0

(a) Rivest

(b) Shamir

(c) Adleman

paper publishing their joint discovery.

So how does it function? First we convert our plaintext into an integer m, which can be done in various different ways called padding schemes. How we do this is not so important, as long as it is public. To form our public key, we choose two randomly generated prime numbers, p and q, which should be around 300 digits long each, and a small integer e. The product $n = pq$ and e are released publicly, but not the primes themselves.

We generate a ciphertext c, using $c \equiv m^e \mod n$.

To revert from ciphertext c back to m we use Euler's theorem that $m^{\varphi(n)} \equiv 1 \mod n$, which implies that $m^{k\varphi(n)+1} \equiv m \mod n$.

All we need is to find d such that $ed = k\varphi(n) + 1$. Then $c^d \equiv m^{ed} \equiv m \mod n$. How can we find d? Using Euclid's algorithm of course! This d is our private key, and cannot be computed without knowing the value of φn. If you know the primes you can simply compute $\varphi(n) = (p-1)(q-1)$ as shown earlier, but if you don't know the primes it's impossible, so they are a *trapdoor*!

This is all rather complicated, so let's work through an example with small numbers. Say we pick $p = 53,\ q = 97,\ e = 49$. We compute $n = 53 \times 97 = 5141$.
$\varphi(n) = (97-1)(53-1) = 4992$ The private key will be an integer solution to the equation

$49d - 4992k = 1$. We find d using Euclid's algorithm as follows:

$$\begin{aligned}
4992 &= 101 \times 49 + 43 \\
49 &= 1 \times 43 + 6 \\
43 &= 7 \times 6 + 1 \\
1 &= 43 - 7 \times 6 \\
&= (49 - 6) - 7(49 - 43) \\
&= (49 - (49 - 43)) - 7(49 - (4992 - 101 \times 49)) \\
&= (49 - (49 - (4992 - 101 \times 49))) - 7(49 - (4992 - 101 \times 49)) \\
&= (49 - (-4992 + 102 \times 49)) - 7(-4992 + 102 \times 49) \\
&= (4992 - 101 \times 49) + 7 \times 4992 - 714 \times 49 \\
&= 8 \times 4992 - 815 \times 49
\end{aligned}$$

This gives us one solution ($d = -815,\ k = 8$) to the equation, but we want a solution with a positive value of d, so we add 4992 to d and subtract 49 from k, at last giving us our private key $d = 4177$. If we receive an encrypted message c, we recover the original message m by reducing $c^{815} \mod 5141$. This looks hard though, as a number to the power of 815 will be absolutely huge. We can make this easier computationally by doing it incrementally. Take $c = 11$, for example. We first reduce (using a computer obviously)

$$11^4 \equiv 14641 \equiv 4359 \mod 5141$$

Then we continue

$$\begin{aligned}
11^8 &\equiv (11^4)^2 \equiv 4359^2 \equiv 4886 \mod 5141 \\
11^{16} &\equiv (11^8)^2 \equiv 4886^2 \equiv 3333 \mod 5141 \\
&\vdots
\end{aligned}$$

... and so on. We can repeat this process and compute $11^{815} \mod 5141$ as a product of these, which will provide us with the original message (4064), which we would then have to convert into plaintext using the public padding scheme.

We've done a whole lot of maths. But why does it matter? Aside from the security of RSA, there's another advantage it has over conventional ciphers: signatures.

When you receive a message, how can you tell who it came from? In standard cryptography, there's always the threat of a third party who has discovered the encryption key sending communications pretending to be someone else to extract information. This is even more problematic for public-key cryptography, as anyone can send an encrypted message using the public key, but it's an issue RSA solves.

Say I want to send a message to Dr Crawford. First, I choose a **secure hash function** notated hash(x), which is like a trapdoor function without a trapdoor: there's no known inverse. I encrypt my plaintext m using Tom's public key. I also calculate $hash(m)$ and encrypt this using my **private key**. I send both to Tom, who decrypts the first with his private key and the second with **my** public key. As he now has my plaintext, he can compute $hash(m)$ and compare this with the second part of my message.

If they are the same, he knows the message is from me, and if they are different, he knows someone is pretending to be me or the message has been altered in transit. Because altering the cipher would change the hash value of the plaintext, altering it would put it out of line with the second value, revealing that a third party was at play!

Now to end off: a treat for the curious readers in the audience. If you found this interesting you should have a deeper look into secure hash functions, and how they can be used to secure cryptocurrencies like Bitcoin, as well as elliptic-curve cryptography, another asymmetric cipher using a more intricate trapdoor.

I think this illustrates the value of seemingly abstract mathematics. The world is complicated, and we often want to do complicated things (like cryptography!) in it. The mathematical theories underpinning these complexities may often seem disconnected from the real world, but they're incredibly important.