In what follows it will be convenient to have an efficient way to calculate powers of an integer modulo n.

Given x, n, and e, we wish to calculate $x^e \mod n$. Fast exponentiation proceeds as follows. Write

$$e = \sum_{i=0}^{k} e_i 2^i$$

where $0 \leq e_i \leq 1$. (This is just the binary expansion of e.) Then

$$x^e = x^{\sum_{i=0}^{k} e_i 2^i} = \prod_{\substack{0 \leq i \leq k \\ e_i = 1}} x^{2^i}.$$

Now calculate the x^{2^i} by repeatedly squaring, and multiply together as required.

Example 3.17: Compute $7^{89} \mod 17$.

$$89 = 64 + 16 + 8 + 1 = 2^6 + 2^4 + 2^3 + 2^0.$$

We have

$$\begin{aligned} 7^2 &\equiv 49 \equiv 15 \mod 17 \\ 7^{2^2} &\equiv 225 \equiv 4 \mod 17 \\ 7^{2^3} &\equiv 16 \equiv 16 \mod 17 \\ 7^{2^4} &\equiv 256 \equiv 1 \mod 17 \\ 7^{2^5} &\equiv 7^{2^6} \equiv 1 \mod 17 \end{aligned}$$

So

$$\begin{aligned} 7^{89} &\equiv 7 \times 7^{2^3} \times 7^{2^4} \times 7^{2^6} \\ &\equiv 7 \times 16 \times 1 \times 1 \equiv 10 \mod 17. \end{aligned}$$

Here we calculated 9 products instead of 88.

4. Public Key Cryptography

Recall the various levels of security that a cipher might possess. So far we have looked at examples of ciphers of various forms, but all of them can be deciphered by anyone who knows the system and the key. But for our highest level of security we did not want this to be possible.

Suppose that we operate a bank with millions of customers, all of whom communicate with us electronically. We would like to communicate securely, but do not have the time or resources to set up individual secret ciphers with each customer. To do this we would need to verify the identity of each one, and then exchange keys securely.

It would be much easier if we could give all of our customers the same instructions on how to encode their messages. They can then send us the encrypted message and we can decrypt it. However, this relies on having a cipher where it is possible to encrypt without being able to decrypt. Otherwise Eve (who could be another customer) can decrypt all messages as well.

Such encryption systems do exist, and are examples of public key cryptography. The idea is to give out a public key which is known by everybody, and which is used to encode the message, but keep secret a private key which is needed to decrypt the message.

How can we construct such systems? We need a function $f : \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$ which can be easily computed, but for which the inverse function is almost impossible to calculate without extra information. The function f, and n, will be our public key, and the extra information needed for the inverse will be our private key.

We will consider two famous examples of public key systems: the RSA algorithm (due to Rivest, Shamir, and Adleman), and ElGamal encryption (which is based on the Diffie-Helman protocol).

Ciphers where the encryption key needs to be kept secret between the sender and recipient provide private key cryptography. These can be more efficient to use than public key systems. In such cases Alice and Bob may choose to use a public key encryption to share their key, and then use this secret key as the basis of their private key encryption. Diffie-Helman is one procedure for doing this.

The RSA system relies on the difficulty of factorising a number into a product of primes. The basic idea is quite simple.

Bob generates randomly two large prime numbers p and q, and chooses e such that

$$1 < e < \phi(pq) \quad \text{and} \quad (e, \phi(pq)) = 1.$$

(Recall that $\phi(pq) = (p-1)(q-1)$ by Corollary 3.16.)

Now set $n = pq$. The numbers n and e are the public key and are available to everybody.

Bob also needs a private key. He computes $1 < d < \phi(pq)$ such that

$$de \equiv 1 \mod \phi(pq).$$

We know that such a d exists (as $(e, \phi(pq)) = 1$) by Theorem 2.18. We also know how to compute it using the extended Euclidean algorithm.

The idea is that anyone can encrypt a message using n and e, but that it can only be decrypted if d is known. And it can be shown that knowing d is essentially equivalent to knowing p and q. Thus if factorising large numbers is hard, then RSA should be a secure system.

But we still need to know how to encrypt and decrypt.

Encryption: Given an integer m with $0 \leq m < n$ we encrypt m by setting

$$c = m^e \mod n.$$

Alice knows n and e, so can encrypt m into c. This is sent to Bob.

Decryption: We claim that Bob can easily decrypt the message, indeed that

$$m = c^d \mod n.$$

This follows from

Theorem 4.1: Let n, e, and d be chosen as above. Then

$$(m^e)^d \mod n = m$$

for all $0 \leq m < n$.

Proof: We know by assumption that $ed \equiv 1 \mod \phi(n)$. Therefore

$$ed = 1 + u\phi(n)$$

for some $u \in \mathbb{Z}$. Now

$$(m^e)^d = m^{ed} = m^{1+u\phi(n)} = m(m^{u(p-1)(q-1)}).$$

Claim:

$$m(m^{u(p-1)(q-1)}) \equiv m \mod p.$$

This is obvious if p divides m as then both sides equal $0 \mod p$. If p does not divide m then $m^{p-1} \equiv 1 \mod p$ by Euler's Theorem (3.7), and so

$$m(m^{u(p-1)(q-1)}) \equiv m(1^{u(q-1)}) \equiv m \mod p.$$

By the same argument $m(m^{u(p-1)(q-1)}) \equiv m \mod q$.

Let $t = (m^{u(p-1)(q-1)}) - m$. We have just shown that

$$t \equiv 0 \mod p \quad \text{and} \quad t \equiv 0 \mod q.$$

As $(p, q) = 1$ we have by Lemma 3.10 that $t \equiv 0 \mod pq$, i.e.

$$m(m^{u(p-1)(q-1)}) \equiv m \mod n.$$

So $(m^e)^d \equiv m \mod n$. But the only number $x < n$ with $x \equiv m \mod n$ is $x = m$, and hence

$$(m^e)^d \mod n = m.$$

□

Example 4.2: Let $p = 17$ and $q = 11$. Then $n = pq = 187$ and $\phi(n) = (p-1)(q-1) = 160$.

We pick $1 < e < 160$ with $(e, 160) = 1$, for example $e = 3$. Next we need to find $1 < d < 160$ with

$$ed \equiv 1 \mod 160.$$

Using the extended Euclidean algorithm we see that $d = 107$.

Now let us encode the message $m = 6$.

$$m^e \mod n = 6^3 \mod 187 = 216 \mod 187 = 29.$$

So the ciphertext is $c = 29$.

To decrypt we need to calculate $29^{107} \mod 187$. We have

$$107 = 64 + 32 + 8 + 2 + 1 = 2^6 + 2^5 + 2^3 + 2^1 + 2^0.$$

Now

$$\begin{aligned}
29^{2^1} &\equiv 29^2 \equiv 841 \equiv 93 \mod 187\\
29^{2^2} &\equiv 93^2 \equiv 8649 \equiv 47 \mod 187\\
29^{2^3} &\equiv 47^2 \equiv 2209 \equiv 152 \mod 187\\
29^{2^4} &\equiv 152^2 \equiv (-35)^2 \equiv 103 \mod 187\\
29^{2^5} &\equiv 103^2 \equiv 10609 \equiv 137 \mod 187\\
29^{2^6} &\equiv 137^2 \equiv (-50)^2 \equiv 69 \mod 187
\end{aligned}$$

So

$$\begin{aligned}
29^{107} &\equiv 29 \times 93 \times 152 \times 137 \times 69 \mod 187\\
&\equiv 2697 \times 20,824 \times 69 \mod 187\\
&\equiv 79 \times 67 \times 69 \mod 187\\
&\equiv 6 \mod 187
\end{aligned}$$

and hence we have indeed recovered the original message.

Our second public key system relies on the difficulty of calculating discrete logarithms.

Let p be a prime. Then it is not hard to show that there exists an element $g \in \mathbb{Z}_p$ such that every other non-zero element of $\mathbb{Z}_p$ is a power of g. Such an element g is called a primitive root of $\mathbb{Z}_p$. (Primitive roots are not unique.)

Given $1 \leq x \leq p-1$, there exists $0 \leq a \leq p-2$ such that

$$x \equiv g^a \mod p.$$

The number a is called the discrete logarithm of x to base g in $\mathbb{Z}_p$. No efficient algorithm is known for computing discrete logarithms.

Example 4.3: For $p = 7$ the element $g = 3$ is a primitive root. For

$$\begin{aligned}
g^1 &\equiv 3 \mod 7\\
g^2 &\equiv 9 \equiv 2 \mod 7\\
g^3 &\equiv 2 \times 3 \equiv 6 \mod 7\\
g^4 &\equiv 6 \times 3 \equiv 18 \equiv 4 \mod 7\\
g^5 &\equiv 4 \times 3 \equiv 12 \equiv 5 \mod 7\\
g^6 &\equiv 5 \times 3 \equiv 15 \equiv 1 \mod 7
\end{aligned}$$

So the discrete logarithm of 6 to base 3 in $\mathbb{Z}_7$ is 3, and the discrete logarithm of 5 to base 3 in $\mathbb{Z}_7$ is 5.

Suppose that Alice and Bob wish to use an encryption system where the key must be kept secret. How can they agree on such a key without Eve finding out? One method is Diffie-Helman key exchange.

Alice and Bob agree on a prime p and an element g with $2 \leq g \leq p-2$, such that g has many distinct powers modulo p. (For example, g could be a primitive root.) This information is public knowledge.

Next Alice chooses some $0 \leq a \leq p-2$ and computes

$$A = g^a \mod p.$$

The number a is secret, but A is sent to Bob (and hence is public).

Bob chooses some $0 \leq b \leq p-2$ and computes

$$B = g^b \mod p.$$

The number b is secret, but B is sent to Alice (and hence is public).

Now Alice computes

$$B^a \mod p = g^{ab} \mod p$$

and Bob computes

$$A^b \mod p = g^{ab} \mod p.$$

Therefore Alice and Bob both now know $g^{ab} \mod p$, and they can use this as their secret key for whichever cryptosystem they like best.

Eve knows A, B, g, and p, but needs to know $g^{ab} \mod p$. The only known method for finding this involves using discrete logarithms to compute a from A and b from B. This is believed to be a hard problem.

There is a related public key system called ElGamal encryption. Alice chooses p and a as above, and a primitive root g of $\mathbb{Z}_p$, and calculates A as before. Her public key is (p, g, A).

To send a message m with $0 \leq m \leq p-1$, Bob chooses b and calculates B as before. Finally Bob calculates

$$c = A^b m \mod p$$

and sends the pair (B, c) to Alice.

Alice needs to be able to decrypt the message. She does this as follows. Let $x = p-1-a$. Then

$$\begin{aligned} B^x c &\equiv g^{b(p-1-a)} A^b m && \mod p \\ &\equiv (g^{p-1})^b (g^a)^{-b} A^b m && \mod p \\ &\equiv 1^b (g^a)^{-b} A^b m && \mod p \quad \text{(by Euler's theorem)} \\ &\equiv A^{-b} A^b m && \mod p \\ &\equiv m && \mod p. \end{aligned}$$

Hence Alice can recover the original message m.

Example 4.4: Let $p = 31$ and $g = 17$. Alice chooses $a = 5$ and calculates

$$A = 17^5 \mod 31 = 26.$$

Bob chooses $b = 11$ and calculates

$$B = 17^{11} \mod 31 = 22.$$

For Diffie-Helman, Alice calculates

$$B^a \mod p = 22^5 \mod 31 = 6$$

and Bob calculates

$$A^b \mod p = 26^{11} \mod 31 = 6.$$

Thus they both now know the secret key is 6.

For ElGamal, Bob also calculates

$$c = A^b m \mod p.$$

If $m = 11$ then

$$c = 6 \times 11 \mod 31 = 4.$$

To decode, Alice calculates

$$B^{p-1-a} c = 22^{25} \times 4 \mod 31 = 11$$

and hence recovers the original message $m = 11$.

Let us return to the RSA algorithm. We saw that to decrypt a message we need to calculate $c^d \mod n$. If we choose e to be small (so that encoding is easy) then d will have to be large, as

$$de \equiv 1 \mod (p-1)(q-1)$$

implies that $de > (p-1)(q-1)$. But then calculating $c^d \mod n$ will be (relatively) slow.

We can speed up decryption by using the Chinese Remainder Theorem. Recall that Bob knows both p and q. On receiving the ciphertext c he calculates the following.

Let

$$d_1 = d \mod p-1 \quad \text{and} \quad d_2 = d \mod (q-1).$$

Set

$$m_1 = c^{d_1} \mod p \quad \text{and} \quad m_2 = c^{d_2} \mod q.$$

As p and q are much smaller than pq these numbers will be easy to calculate.

Note that Euler's Theorem (3.7) implies that

$$m_1 \equiv c^d \mod p$$

as $d = a(p-1) + d_1$ for some $a \in \mathbb{Z}$ and $c^{p-1} \equiv 1 \mod p$ when c is not a multiple of p. Similarly we have

$$m_2 \equiv c^d \mod q.$$

Now use the Chinese Remainder Theorem to calculate $0 \leq m \leq n-1$ such that

$$m \equiv m_1 \mod p \quad \text{and} \quad m \equiv m_2 \mod q.$$

Recall that to find m we use the extended Euclidean algorithm to find $y_1, y_2 \in \mathbb{Z}$ with

$$y_1 p + y_2 q = 1$$

and set

$$m = m_1 y_2 q + m_2 y_1 p \mod n.$$

Then m is the unique solution modulo n of the pair of congruences defining m_1 and m_2. But clearly $c^d \mod n$ is another such solution, and hence $m = c^d \mod n$, which is the desired plaintext.

Example 4.5: Consider the message we sent using RSA in Example 4.2. We had $p = 17$, $q = 11$, $n = 187$, and $e = 3$, $d = 107$. The plaintext was $m = 6$ and the ciphertext was $c = 29$.

To decrypt we had to calculate $29^{107} \mod 187$. Using the Chinese Remainder Theorem method we have

$$d_1 = 107 \mod 16 = 11 \quad \text{and} \quad d_2 = 107 \mod 10 = 7.$$

Now

$$m_1 = 29^{11} \mod 17 = 12^{11} \mod 17 = 8^5 \times 12 \mod 17 = 6$$

and

$$m_2 = 29^7 \mod 11 = 7^7 \mod 11 = 6.$$

In this example it is clear that a solution of the equations

$$m \equiv 6 \mod 17 \quad \text{and} \quad m \equiv 6 \mod 11$$

is just $m = 6$, and hence this is the desired answer. However in general the final 2 equations will involve different values, and so we will have to use the extended Euclidean algorithm to solve them.

For large values of p and q and d, this new method of decryption is considerably more efficient than before.

We have seen two public key cryptosystems, RSA and ElGamal. How secure are they?

RSA relies on the difficulty of factorising large numbers. It can be shown that if d and e (and n) are known, then it is possible to efficiently factorise $n = pq$. Thus if Eve can find a way to calculate d, then she can factorise large numbers.

However, it is *not* known if it is possible to crack RSA without first calculating d.

Similarly, ElGamal relies on the difficulty of calculating discrete logarithms.

It may be that one day someone finds an efficient method of factorising, or of calculating discrete logarithms. If they do then these two ciphers will become insecure. Thus the search for new and more secure ciphers continues.

The RSA trio issued a challenge in 1977 to decrypt a message encoded using RSA, where n was a 129 digit number and $e = 9007$. They thought that it would be far too hard to crack in their lifetime.

However in 1994 it was cracked after 8 months of computation using a method which split the problem up so it could be tackled by a few thousand computers at the same time. This illustrates why the search for stronger ciphers is an important practical problem.

It is worth noting that Rivest estimated that it would take 4×10^{13} years of computer time to crack this example, even if each multiplication could be carried out in a nanosecond. This speed is still not possible, but the problem was still solved in far less time than predicted. Thus it is not just because computers have become faster than ciphers can be broken; our methods of breaking such ciphers have also improved.

Some people are trying to develop quantum computers. These are theoretically possible, but don't yet exist for practical purposes. Such a computer would make *all* current cryptosystems vulnerable to cracking. However, such a computer could also be used with quantum cryptosystems to send messages which are *impossible* to crack.