

# Divisibility, Modular Arithmetic, and RSA

## RSA Outline

By an encryption scheme, we mean a pair of algorithms  $E$  (encoding) and  $D$  (decoding) with the following relationship.

$$\begin{array}{ccccccc} \text{message} & \text{encryption} & \text{encrypted message} & \text{decryption} & \text{original message again} & & \\ M & \rightarrow & C = E(M) & \rightarrow & M = D(C) = D(E(M)) & & \end{array}$$

In a **private key** encryption scheme, the sender and recipient agree on some information (secretly) that are used to encrypt and decrypt. Since no one else (presumably) knows this information, only the sender and receiver can encrypt and decrypt. The Caesar cypher is a trivial example of private key encryption.

RSA is a **public-key** encryption scheme. In a public key encryption scheme, there are two categories of information used in the encryption/decryption algorithms:

- public information:
  - Freely available (to sender, recipient and eaves droppers).
  - Different for each recipient.
  - Senders of messages use this information to encrypt.
- secret information:
  - Only the recipient knows it (recipient hopes).
  - Recipient uses this to decrypt.

The scheme is amazingly easy to describe.

1. Public information: integers  $n$  and  $e$
2. Private information: integer  $d$
3. Encryption:  $C = E(M) = M^e \pmod{n}$
4. Decryption:  $D(C) = C^d \pmod{n}$

Here we are assuming that the text message has been parsed into “blocks” and that each block is coded as a number mod  $n$ . ( $n$  will be large, say around 400 bits.)

Of course, not just any  $n$ ,  $d$ , and  $e$  will do. We need to choose them in such a way that

1. Decryption works:  $M = D(E(M))$ . That is,  $(M^e)^d \equiv M \pmod{n}$ .
2. The scheme is secure: It is not easy to determine  $d$  if you know  $e$  and  $n$ .
3. The scheme is practical: Suitable  $e$ ,  $d$ , and  $n$  can be found (efficiently), and the algorithms  $D$  and  $E$  can be performed efficiently.

Number theory (modular arithmetic) is what allows us to make all of this work.

# The Number Theory

Unless otherwise stated, all the variables below are integers.

## Divisibility

1. Definition:  $a \mid b$  means  $a$  divides  $b$  (evenly), that is, there is an integer  $k$  such that  $ka = b$ .
2. **Divisibility Lemma:** The following basic facts are all straightforward to prove from definition, but are useful enough to collect into a lemma.
  - (a) If  $a \mid x$  and  $a \mid y$ , then  $a \mid x \pm y$ .
  - (b) If  $a \mid x$  and  $a \mid x \pm y$ , then  $a \mid y$ .
  - (c) If  $x \mid y$  and  $y \mid z$ , then  $x \mid z$ .
  - (d) If  $a \mid x$ , then  $a \mid kx$  for any integer  $k$ .

The first two of these were the key to **Euclid's GCD Algorithm**.

## Modular Arithmetic

1. Definition of mod operator  
If  $a \nmid b$ , then when we divide  $b$  by  $a$  there is a remainder left over. That is
$$b = aq + r \quad r \in [0, a - 1]$$
Then  $b \bmod a = r$  (the remainder when dividing  $b$  by  $a$ ).
2. Definition of mod relation  
We say that  $a \equiv b \pmod{n}$  if  $n \mid b - a$ . The following things are all equivalent:
  - (a)  $a \equiv b \pmod{n}$
  - (b)  $n \mid b - a$
  - (c)  $a - b = nk$  for some integer  $k$
  - (d)  $a = nk + b$  for some integer  $k$
  - (e)  $a \bmod n = b \bmod n$ .
3. Modular Arithmetic: If  $a \equiv x \pmod{n}$  and  $b \equiv y \pmod{n}$ , then
  - (a)  $a + b \equiv x + y \pmod{n}$
  - (b)  $a - b \equiv x - y \pmod{n}$
  - (c)  $a \cdot b \equiv x \cdot y \pmod{n}$

So we can do modular arithmetic by taking any member of a congruence class.

### 4. Inverses.

The inverse of  $a \pmod{n}$  is an integer  $x$  such that  $ax \equiv 1 \pmod{n}$ . There is not always an inverse. 0 never has an inverse (because  $0 \cdot x = 0 \not\equiv 1 \pmod{n}$ ). But other inverses also might fail to exist. Fortunately, it is easy to determine if there is an inverse and to find it if it exists:

- (a)  $a$  has an inverse  $\pmod{n}$  if and only if  $\gcd(a, n) = 1$ .
- (b) **GCD Lemma:** For any  $a$  and  $n$ , we can always find integers  $s$  and  $t$  such that

$$\gcd(a, n) = sa + tn$$

by running **Euclid's GCD Algorithm** and backsubstituting for each remainder.

(c) If  $\gcd(a, n) = 1$ , this gives

$$1 = sa + tn$$

from which we see that  $s$  is an inverse of  $a \pmod{n}$  (and  $t$  is an inverse of  $n \pmod{a}$ ). To see this, just interpret the equation mod  $n$  or mod  $a$ .

(d) Division: Inverses allow us to do a sort of integer division (division equals multiplication by inverse).

(e) Linear equations. This allows us to solve linear equations like  $ax \equiv b \pmod{n}$  whenever  $a$  has an inverse mod  $n$ . (To solve, multiply both sides by the inverse of  $a$  and simplify.)

## A bit more advanced Number Theory

We can use the tools above to show a few more advanced things that will be useful for RSA.

1. **First Cancellation Lemma.** If  $\gcd(a, c) = 1$  and  $a \mid bc$ , then  $a \mid b$ .

Reason: Write  $1 = sa + tc$  (gcd lemma). Then  $b = sab + tcb$ .  $a \mid sab$ , and  $a \mid tcb$  (because  $a \mid bc$ ), so  $a \mid b$  (divisibility lemma).

2. **Second Cancellation Lemma.** If  $\gcd(n, c) = 1$  and  $ac \equiv bc \pmod{n}$ , then  $a \equiv b \pmod{n}$ .

Reason:  $n \mid ac - bc$ , so  $n \mid c(a - b)$ , so  $n \mid (a - b)$  (cancellation lemma), so  $a \equiv b \pmod{n}$ .

Alternative Reason:  $c$  has an inverse mod  $n$ . Call it  $d$ . Then  $a \equiv acd \equiv bcd \equiv b \pmod{n}$ .

3. **Prime Divisibility Lemma:** If  $p \mid a_1 \cdot a_2 \cdots a_n$ , then  $p \mid a_i$  for some  $i$

Reason: repeated use of First Cancellation Lemma.

4. **Unique Factorization.** Every integer can be factored as a product of primes in exactly one way (up to the order of the primes involved).

Reason: Show that any two factorizations are the same because “everything cancels”. (Uses Cancellation Lemma and Prime Divisibility Lemma.)

5. **Chinese Remainder Theorem.** Let  $m_1, m_2, \dots, m_k$  be pairwise relatively prime positive integers, and let  $a_1, a_2, \dots, a_k$  be any integers. Then there is a number  $x$  such that  $x \pmod{m_i} = a_i$  for each  $i$ . (i.e.,  $x \equiv a_k \pmod{m_k}$  for each  $k$ .) In fact, there is always exactly one such number in the range  $[0, M - 1]$ , where  $M$  is the product of all the  $m_i$ 's

(a) Example: Find  $x$  so that  $x \pmod{3} = 1$ ,  $x \pmod{5} = 2$ , and  $x \pmod{8} = 3$ .

Solution: The idea is to fill in the following boxes with integers:

$$x = \square \cdot 40 + \square \cdot 24 + \square \cdot 15$$

where  $40 = 5 \times 8$ ,  $24 = 3 \times 8$ , and  $15 = 3 \times 5$  (products of all but one modulus).

The value in the first box will determine  $x \pmod{3}$ , because the rest is divisible by 3; the value in the second box will determine  $x \pmod{5}$  because the rest is divisible by 5; the value in the third box will determine  $x \pmod{8}$  because the rest is divisible by 8.

To get the appropriate values for each box, we solve the equations  $B_1 40 \equiv 1 \pmod{3}$ ,  $B_2 24 \equiv 1 \pmod{5}$ ,  $B_3 15 \equiv 1 \pmod{8}$ , which we can do using inverses.

(b) This method will always work if the moduli are pairwise relatively prime, because in that case the required inverses will exist.

6. **Fermat's Little Theorem.** If  $p$  is a prime and  $a \not\equiv 0 \pmod{p}$ , then  $a^{p-1} \equiv 1 \pmod{p}$  and  $a^p \equiv a \pmod{p}$ .

Basic idea: There will be a number  $a$  such that the numbers  $a, a^2, a^3, \dots, a^{p-1} \pmod{p}$  are all distinct and not 0, so 1 can't occur in the list until all the other non-zero numbers  $\pmod{p}$  have occurred. (Proving such an  $a$  always exists is the tricky part.) This means that  $a^{p-1} \equiv 1$ . For other numbers  $b$ ,  $b \equiv a^d$  for some  $d$ , so  $b^{p-1} \equiv (a^d)^{p-1} \equiv (a^{p-1})^d \equiv 1^d \equiv 1$ .

## Showing that RSA decoding works

### Choosing the parameters: $n$ , $e$ , and $d$

Here's how we choose  $n$ ,  $e$ , and  $d$ :

- First choose two large primes  $p$  and  $q$ . (Keep these secret.)
- Let  $n = pq$ .
- Choose  $e$  and  $d$  so that they are inverses mod  $(p-1)(q-1)$ . (Note that  $e$  has an inverse  $d$  if  $\gcd(e, (p-1)(q-1)) = 1$ . There will be many such  $e$ .)

So why does this work? Remember the goal: we want  $(M^e)^d \equiv M \pmod{n}$ . Let's look at this more closely.  $(M^e)^d \equiv M^{ed}$ , so we want  $M^{ed} \equiv M$ . This follows from  $M^{ed-1} \equiv 1$ , so let's see how to achieve that.

Since  $\gcd(p, q) = 1$ , by the Chinese Remainder Theorem we can break this into two pieces:  $M^{ed-1} \equiv 1 \pmod{p}$  and  $M^{ed-1} \equiv 1 \pmod{q}$ .

By Fermat's Little Theorem,  $M^{p-1} \equiv 1 \pmod{p}$  and  $M^{q-1} \equiv 1 \pmod{q}$ . Since  $e$  and  $d$  are inverses mod  $(p-1)(q-1)$ , we can write  $ed = 1 + k(p-1)(q-1)$ . So  $M^{ed-1} \equiv M^{k(p-1)(q-1)} \equiv (M^{p-1})^{k(q-1)} \equiv 1^{k(q-1)} \equiv 1 \pmod{p}$ . By a similar argument  $M^{ed-1} \equiv 1 \pmod{q}$ , hence  $M^{ed-1} \equiv 1 \pmod{n}$ , so  $M^{ed} \equiv 1M \equiv M \pmod{n}$ .

### A few comments on RSA

1. Finding large primes  $p$  and  $q$  is reasonably easy.

About 1 in every 460 200-digit numbers is a prime. There are randomized methods to check whether a number is prime that work most of the time, so if we try 1000 or so 200-digit numbers, we will probably have found 2 primes. This cost is a one-time off-line cost of using RSA

2.  $M^e \pmod{n}$  can be computed reasonably efficiently.

The obvious algorithm (raise  $M$  to the  $e$ th power by repeated multiplication) is terrible: it takes about  $10^{400}$  steps if  $e$  is a 400-digit number! But repeated squaring (doing the modular arithmetic as we go) is much better:

$$M^e = M^{e/2} M^{e/2} \quad \text{or} \quad M^e = M^{e/2} M^{e/2} M$$

This requires only about  $\log_2 e$  steps to compute (i.e., only a few hundred steps for 400-digit numbers – in general, linear in the length of  $e$ ). It can be implemented recursively or iteratively. Figuring out which case is easy from the binary representation of  $e$ .

So RSA is reasonably efficient to use.

3. RSA is not nearly as efficient to use as some private key systems.

One use of RSA is to communicate a private key pair over an insecure channel, after which the two parties can use a faster private-key system like DES.

4. No method is (publicly) known that can determine  $d$  from  $e$  and  $n$  that does not easily yield a factorization of  $n$ . So RSA seems to be about as hard to crack as factoring large numbers. But there is no mathematical proof that we won't eventually be able to factor efficiently.

This means that RSA is reasonably secure: no one who will admit it knows a good algorithm for factoring 400-digit numbers. (Of course, the size bound here keeps growing as both machines and algorithms improve.)

5. RSA can also be used to do electronic signatures.

To confirm you authored something, code with your decoding algorithm (using  $d$ ) to encode your message (or just a signature). The recipient can decode with  $e$ . If you are the only one who has access  $d$ , then you are the only one who could have sent the message.

This works because  $(M^e)^d = M^{ed} = (M^d)^e$ .