1 Modern Cryptography

(Continued from previous notes.)

1.1 Interlude: Binary Exponentiation

With Euler's theorem in mind, we can compute large powers in modular arithmetic very quickly. Let's now explain the process more systematically. Suppose we have a positive integer n and an integer a with $\gcd(a, n) = 1$, and that we want to compute $a^m \pmod{n}$ for some large number n.

The first step is to calculate $r = m \pmod{\phi(n)}$. If we do this, we'll have $m = \phi(n)q + r$, so

$$a^m = a^{\phi(n)q+r} = (a^{\phi(n)})^q a^r \equiv 1^q a^r = a^r \pmod{n}. \tag{1}$$

from Euler's Theorem and the Modular Arithmetic Theorem. This immediately reduces the power we have the compute substantially. From there, we can compute $a^r \pmod{n}$ using a technique called **binary exponentation.** The idea is fairly straightforward in examples, so let's look at an example.

(Example.) Suppose we find that $r = 25$. How can we compute $a^r \pmod{n}$? One thing we could do is multiply a by itself mod n repeatedly, but this would require 25 multiplications mod n. Here's another thing we can try; we square repeatedly:

- Find $a^2 \pmod{n}$.
- Find $a^4 = (a^2)^2 \pmod{n}$ by squaring the result of the previous step.
- Find $a^8 = (a^4)^2 \pmod{n}$ by squaring the result of the previous step.
- Find $a^{16} = (a^8)^2 \pmod{n}$ by squaring the result of the previous step.

Squaring this again would go past $r = 25$, so we stop at 16. We now want to figure out how to find a^{25} using the powers of a that we computed already. Notice that

$$25 = 16 + 8 + 1$$

so

$$a^{25} = a^{16+8+1} = a^{16}a^8a^1,$$

and we already know $a^{16}, a^8, a^1 \pmod{n}$, so we can compute $a^{25} \pmod{n}n$ by multiplying these three values together mod n. This only requires 6 multiplication in total, much less than 25.

(Example.) Suppose we want to compute $3^{4398391} \pmod{80}$. First, note that $\phi(80) = 32$ and $4398391 \equiv 23 \pmod{32}$, so $3^{4398391} \equiv 3^{23} \pmod{80}$ by Euler's Theorem. Now, using binary exponentation, we have

$$\begin{aligned}
3^2 &= 9 \\
3^4 = (3^2)^2 &= 9^2 = 81 \equiv 1 \pmod{80} \\
3^8 = (3^4)^2 &\equiv 1^2 = 1 \pmod{80} \\
3^{16} = (3^8)^2 &\equiv 1^2 = 1 \pmod{80} \\
3^{23} = 3^{16+4+2+1} &= 3^{16}3^43^23 \equiv 1 \cdot 1 \cdot 9 \cdot 3 = 27 \pmod{80}.
\end{aligned}$$

(Exercise.) Compute $3^{293423948903859017} \pmod{50}$.

Note that $\phi(50) = 20$ and $293423948903859017 \equiv 17 \pmod{20}$, so

$$3^{293423948903859017} \equiv 3^{17} \pmod{50}.$$

Using binary exponentation, we have

$$\begin{aligned} 3^2 &= 9 \\ 3^4 &= (3^2)^2 = 9^2 = 81 \equiv 31 \pmod{50} \\ 3^8 &= (3^4)^2 = 31^2 \equiv 11 \pmod{50} \\ 3^{16} &= (3^8)^2 = 11^2 = 121 \equiv 21 \pmod{50}. \end{aligned}$$

From this, $17 = 16 + 1$ and so

$$3^{17} = 3^{16+1} = 3^{16}3^1 \equiv 21 \cdot 3 = 13 \pmod{50}.$$

The "repeated squaring and then multiply together" part of this process is very closely related to finding binary representations of integers. Here is the definition:

Definition 1.1

Let r be a non-negative integer. The **binary representation** of r is a string $b_k \dots b_1 b_0$ where each b_i is either 0 or 1, and where

$$r = b_0 + b_1 2 + b_2 2^2 + \dots + b_k 2^k.$$

The number b_i is called the ith bit of r. We call b_0 the rightmost bit and b_k the leftmost bit.

For example, the binary representation of 25 is `11001` because

$$1 + 0 \cdot 2 + 0 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 = 1 + 8 + 16 = 25.$$

To *find* the binary representation, consider the following algorithm:

(Algorithm.) Let r be a non-negative integer. To find the binary representation of r, divide r by 2, then divide the quotient by 2, and then divide that quotient by 2, and so forth, until you hit a quotient of 2. The remainders of these divisions are the binary representation, with the last remainder corresponding to the leftmost bit.

Remark: This is an algorithm for humans, not for computers. Computers represent integers in binary form.

(Example.) To calculate the binary representation of 193, we divide 193 by 2, and then repeatedly divide the quotient by 2:

$$\begin{aligned} 193 &= 96 \cdot 2 + 1 \\ 96 &= 48 \cdot 2 + 0 \\ 48 &= 24 \cdot 2 + 0 \\ 24 &= 12 \cdot 2 + 0 \\ 12 &= 6 \cdot 2 + 0 \\ 6 &= 3 \cdot 2 + 0 \\ 3 &= 1 \cdot 2 + 1 \\ 1 &= 0 \cdot 2 + 1 \end{aligned}$$

Since we hit a quotient of 0, we stop dividing. The binary representation is the sequence of remainders we found, with the leftmost bit being the last remainder we found and the rightmost bit being the first remainder we found. In other words, the binary representation of 193 is `11000001`.

(Exercise.) Find binary representations of the following integers.

1. 37

$$\begin{aligned} 37 &= 18 \cdot 2 + 1 \\ 18 &= 9 \cdot 2 + 0 \\ 9 &= 4 \cdot 2 + 1 \\ 4 &= 2 \cdot 2 + 0 \\ 2 &= 1 \cdot 2 + 0 \\ 1 &= 0 \cdot 2 + 1. \end{aligned}$$

The resulting binary string is `100101`.

2. 123

$$\begin{aligned} 123 &= 61 \cdot 2 + 1 \\ 61 &= 30 \cdot 2 + 1 \\ 30 &= 15 \cdot 2 + 0 \\ 15 &= 7 \cdot 2 + 1 \\ 7 &= 3 \cdot 2 + 1 \\ 3 &= 1 \cdot 2 + 1 \\ 1 &= 0 \cdot 2 + 1. \end{aligned}$$

The resulting binary string is `1111011`.

3. 290

$$\begin{aligned} 290 &= 145 \cdot 2 + 0 \\ 145 &= 72 \cdot 2 + 1 \\ 72 &= 36 \cdot 2 + 0 \\ 36 &= 18 \cdot 2 + 0 \\ 18 &= 9 \cdot 2 + 0 \\ 9 &= 4 \cdot 2 + 1 \\ 4 &= 2 \cdot 2 + 0 \\ 2 &= 1 \cdot 2 + 0 \\ 1 &= 0 \cdot 2 + 1. \end{aligned}$$

The resulting binary string is `100100010`.

4. 300

$$\begin{aligned} 300 &= 150 \cdot 2 + 0 \\ 150 &= 75 \cdot 2 + 0 \\ 75 &= 37 \cdot 2 + 1 \\ 37 &= 18 \cdot 2 + 1 \\ 18 &= 9 \cdot 2 + 0 \\ 9 &= 4 \cdot 2 + 1 \\ 4 &= 2 \cdot 2 + 0 \\ 2 &= 1 \cdot 2 + 0 \\ 1 &= 0 \cdot 2 + 1. \end{aligned}$$

The resulting binary string is `100101100`.

(Exercise.) Why must the rightmost bit in the binary representation of an even number must be 0?

Note that 2^i is even if $i > 0$ and odd if $i = 0$. In particular, when $i = 0$, it follows that $2^0 = 1$. But, by using the above "algorithm," we find that the binary representation of 1 is just `1` ($1 = 0 \cdot 2 + 1$). Also note that

- adding two even numbers yields an even number,
- adding an even and an odd number yields an odd number,
- and multiplying a number by an even number yields an even number.

Looking at the formula in (1.1), we notice that all the components aside from b_0 will be even (or 0). The only component that can be odd is b_0, and that's when $b_0 = 1$. So, if $b_0 = 1$, then we know that r must be odd and hence the odd number must have rightmost bit 1. Conversely, when r is even, the rightmost bit must be 0.

We can now state the general fact about binary exponentation below:

Lemma 1.1

Let n be a positive integer, and let $b_k \dots b_1 b_0$ be the binary representation of a non-negative integer r. To compute $a^r \pmod n$ for some integer a, first compute $a^{2^i} \pmod n$ for $i = 0, 1, \dots, k$ by repeated squaring. Then, to get a^r, multiply together all of the 2^{a^i} where $b_i = 1$. In other words,

$$a^r \equiv \prod_{b_i = 1} a^{2^i} \pmod n.$$

1.2 Interlude: Primality Testing

How can we quickly figure out if a number is prime? Remember that factoring is computationally expensive, so it's not a good idea to try to figure out that a number is prime by factoring it! There are a number of tests we can do, although we'll focus on one called the **Miller-Rabin test**. Before we talk about the test, we should establish some important background results.

Lemma 1.2

If n is prime, the only solutions to $x^2 \equiv 1 \pmod n$ are $x \equiv \pm 1 \pmod n$.

Lemma 1.3: Miller-Rabin

Suppose n is a positive *odd* integer and write $n-1=2^s d$ for some positive integer s and some odd number d. Suppose a is an integer between 1 and $n-1$. If n is prime, then one of the following congruence relations must hold true:

$$a^d \equiv 1 \pmod{n}$$
$$a^d \equiv -1 \pmod{n}$$
$$a^{2d} \equiv -1 \pmod{n}$$
$$a^{2^2 d} \equiv -1 \pmod{n}$$
$$\vdots$$
$$a^{2^{s-1} d} \equiv -1 \pmod{n}$$

(Example.) Consider $n=41$, a prime number. We have

$$n-1=40=2^3 5.$$

In other words, we can take $s=3$ and $d=5$ in the above lemma. Fix some integer $a=17$. Since $s=3$, we have 4 congruences in our list, and one of them should then be true, so let's check which one.

$$17^5 \equiv 27 \pmod{41} \neq \pm 1.$$
$$17^{2\cdot 5} = (17^5)^2 \equiv 27^2 \equiv 32 \pmod{41} \neq -1.$$
$$17^{2^2\cdot 5} = (17^{2\cdot 5})^2 \equiv 32^2 \equiv 40 \equiv -1 \pmod{41}.$$

Here, the last congruence is true.

(Exercise.) For each of the following integers n, find the integers s and d such that $n-1=2^s d$, where d is odd.

(a) 43

We have
$$n-1=43-1=42.$$
Trying different combinations of d, we note that
$$42=2^1\cdot 21.$$
So, $d=21$ and $s=1$.

(b) 49

For $n-1=49-1=48$, we have
$$48=2^4\cdot 3$$
so that $s=4$ and $d=3$.

(c) 65

For $n-1=64$, we have

$$64 = 2^6 \cdot 1$$

so that $s=6$ and $d=1$.

Definition 1.2

Suppose n is a positive odd integer and write $n-1=2^s d$ for some positive integer s and an odd number d. Suppose a is an integer between 1 and $n-1$. We say that n is a strong probable prime to base a if one of the following congruences is true:

$$a^d \equiv 1 \pmod{n}$$

$$a^d \equiv -1 \pmod{n}$$

$$a^{2d} \equiv -1 \pmod{n}$$

$$a^{2^2 d} \equiv -1 \pmod{n}$$

$$\vdots$$

$$a^{2^{s-1} d} \equiv -1 \pmod{n}$$

If n is not a *strong probable prime* to base a, then a is called a *witness for the compositness* of a, or a Miller-Rabin witness for a.

With this definition, the Miller-Rabin Lemma states that "every prime number is a strong probable prime to any base." Equivalently, "if n is not a strong probable prime to some base a, then n must be composite."

(Example.) For $n=25$, we see that $n-1=24=2^3\cdot 3$, so we have $s=d=3$. Suppose we choose a base of $a=7$; then,

$$7^3 \equiv 18 \pmod{25} \neq \pm 1$$

$$7^{2\cdot 3} \equiv (7^3)^2 \equiv 18^2 \equiv 24 \equiv -1 \pmod{25} = -1,$$

so this says that 25 is a strong probable prime to base 7.

Now, if we let $a=2$, notice how

$$2^3 = 8 \neq \pm 1.$$

$$2^{2\cdot 3} = (2^3)^2 = 8^2 = 64 \equiv 14 \pmod{25} \neq -1.$$

$$2^{2^2\cdot 3} = (2^{2\cdot 3})^2 \equiv 14^2 \equiv 21 \pmod{25} \neq -1.$$

Thus, this n is not a strong probable prime to base 2, and 2 is a witness to the compositness of 25.

If n is composite, the most of the possible choices of base $2 \leq a \leq n-2$ will in fact be witnesses for the compositness of n. So, if we try several such bases and none of them turn out to be a witness for compositness, then we can be quite sure that n is in fact prime.

(Miller-Rabin Primality Test.) Suppose n is a positive integer. If n is 2, output `True`. Otherwise, if n is even, output `False`. Otherwise, repeat the following step some fixed pre-determined number of times:

- Choose a random base $2 \leq a \leq n-2$. If a is a witness for the compositness of n, output `False`.

If we reach the end without having output `False`, then output `True`.

If we do k repetitions, the probability of a false positive is less than 4^{-k}. This gets small very quickly; for example, if we do just 10 repetitions, the probability of a false positive is about one in a million.

With this algorithm in mind, we can quickly check if a number is prime without having to factor it. This also gives us an algorithm for generating large prime numbers: basically, just keep generating random numbers until we find one that's prime!

(Algorithm for Generating Large Primes.) Let r be a positive integer. To generate an r-bit prime, run the following steps:

- Pick a random odd integer n between 2^{r-1} and $2^r - 1$.
- Check if n is prime.
  - If it is, return n.
  - Otherwise, return to the first step.