Powers in Modular Arithmetic, and RSA Public Key Cryptography

Lecture notes for Access 2010, by Erin Chamberlain and Nick Korevaar.

In our examples so far we've been assigning numbers to each letter of a plaintext and then using modular arithmetic to construct a cipher, number by number (or letter by letter). In practice this just amounts to a (letter) substitution cipher, and so can be broken easily with frequency analysis. What we will do for RSA cryptography (and what has been done in cryptography for a long time before RSA) is to make packets consisting of lots of letters, and encrypt those. In RSA we will use HUGE moduli $N = pq$, which are products of two different prime numbers, and break messages into number packets which are residue numbers of N. Then we'll encrypt each packet using a power function $\mod N$ which permutes the residue numbers , and hope to decrypt it with another power function $\mod N$. For example, if N has at least 13 digits (i.e. $N > 10^{12}$), then you can use the table on page 9 of Tom Davis' notes, *Cryptography*, to convert any 6-character expression into a residue number for N, since each character is represented with two digits.

Example 1. Use the Davis table on page 9 of his notes to convert the two word sentence *Hello there!* into two number packets, each of which is less than 10^{12}.

Remark 1. If you use the encryption function $f(x) = x + a \mod N$ to permute the residue numbers, corresponding to Caesar shifts for number packets, anyone who understands clock arithmetic can deduce the decryption function $g(x) = x - a \mod N$ immediately. Similarly, if you specify the encryption function $f(x) = ax \mod N$, for $gcd(a, N) = 1$, then ACCESS students and other hard-working smart people could use the Euclidean algorithm to quickly find a multiplicative inverse b of a mod N, in order to find the decryption function $g(x) = bx \mod N$. (Or they might use Maple-like software which has a multiplicative inverse command built in.) So neither of these examples is a one-way function, even with big message packets.

We'll understand modular arithmetic power functions for $N = pq$ by first understanding it for $N = p$, a prime. Let's experiment:

Example 2. Let $N = 11$. Let our candidate encryption function be $f(x) = x^2 \mod 11$, with domain the residue numbers $\{0, 1, 2, \ldots, 10\}$, and range also contained in the residue numbers. Complete the table below and explain why this function doesn't have an inverse decryption function:

x	0	1	2	3	4	5	6	7	8	9	10
x^2	0	1	4	9	5						

Example 3. Keeping $N = 11$, show that the function $f(x) = x^3 \mod 11$ does encrypt (permute) the residue numbers:

x	0	1	2	3	4	5	6	7	8	9	10
x^3	0	1	8								

Example 4. We might hope, based on our experiences with addition and multiplication encryption functions, that if our encryption function is a power function $f(x) = x^e \mod N$, then our decryption function is $g(x) = x^d \mod N$, for some power d. For the encryption function in the previous example $e = 3$. We shall now deduce a possible value for the decryption power d: Since $f(2) = 8 \mod 11$, we want $g(8) = 2$, i.e. $8^d \equiv 2 \mod 11$. Compute successive powers of 8 until you are able to solve this equation for d. (Hint: $d = 7$.)

Exercise 1. But we need to check that the decryption power $d = 7$ works for every x in our residue range! Let group number x check that this is so, for the residue number x, with the following exceptions: Since $x = 1$ is immediate, and since we just checked $x = 2$, Group 1 should check $x = 8$ and Group 2 should check $x = 9$. ($x = 10 \equiv -1 \mod 11$ is also easy.) Be clever to minimize your computing!

Exercise 2. Since RSA cryptography uses moduli $N = pq$, where p and q are (HUGE) prime numbers, we'll experiment with small prime numbers $p = 3$, $q = 5$, $N = 15$, and use the $\mod 15$ table of powers below to figure out good and bad encryption powers e. (A good encryption function permutes the residue numbers, so that it has an inverse decryption function.) First, you will have to fill in rows 6 and 7 of the table! The easy way to fill in a row is to multiply previous entries by the residue number for that particular row, $\mod 15$.

Power table, mod 15

power $\rightarrow$	1	2	3	4	5	6	7	8	9	10
residue										
0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1
2	2	4	8	1	2	4	8	1	2	4
3	3	9	12	6	3	9	12	6	3	9
4	4	1	4	1	4	1	4	1	4	1
5	5	10	5	10	5	10	5	10	5	10
6										
7										
8	8	4	2	1	8	4	2	1	8	4
9	9	6	9	6	9	6	9	6	9	6
10	10	10	10	10	10	10	10	10	10	10
11	11	1	11	1	11	1	11	1	11	1
12	12	9	3	6	12	9	3	6	12	9
13	13	4	7	1	13	4	7	1	13	4
14	14	1	14	1	14	1	14	1	14	1

Exercise 3. $f(x) = x^3 \mod 15$ is a good encryption function. What part of the power table confirms this fact? Find a power d so that $g(x) = x^d \mod 15$ is the decryption function for $f(x)$. Use the power table to check your work.

When Decryption Powers Exist, and Finding Them

We've been doing a lot of experimentation with modular arithmetic, which is a great way to get ideas about what might be true. Number theory has been a favorite for many famous mathematicians, and so some of their names are attached to the following important theorems. Perhaps the mathematicians were led to these theorems by their own experimentation. These results from two centuries ago turn out by pure serendipity to be the underpinning of RSA cryptography, an idea which would not have been conceived of before the advent of computers.

Theorem 1 (Fermat's Little Theorem). If p is a prime and if $0 < a < p$ is a residue number, then $a^{p-1} \equiv 1 \mod p$.

Proof. Pick any non-zero residue a as above, and consider the corresponding row of the mod p multiplication table. (You can make this less abstract by using the mod 7 table as an example, see below.) Since a has a multiplicative inverse mod p, $ax \equiv ay$ only when $x \equiv y$. (Why?) Therefore, as in our previous discussion of multiplication tables, the residues across the row, namely the residues of

$$1a, 2a, 3a, ..., (p-1)a$$

must all be different, i.e. a permuation of the non-zero residues $1, 2, ..., (p-1)$. Thus the product of all these terms satisfies

$$(1a)(2a)...(p-1)a \equiv (1)(2)...(p-1) \mod p,$$

$$a^{p-1}(1)(2)...(p-1) \equiv (1)(2)...(p-1) \mod p.$$

Multiply both sides of this equation by the multiplicative inverses of $2, 3, ...(p-1)$, i.e. cancel the term $(2)(3)...(p-1)$ from both sides of the equation. Deduce

$$a^{p-1} \equiv 1 \mod p.$$

□

Example 5. Here's how to illustrate Little Fermat concretely, using $p = 7$. Start with the mod 7 multiplication table, without the zero row and column:

mod 7 multiplication table

$\times$	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

Take any row, say the row for $a = 3$. The entries going across are the residues for

$$(3)(1), (3)(2), (3)(3), (3)(4), (3)(5), (3)(6)$$

and they are just a permuation of the original non-zero residues. Thus, taking the product of the entries in this row, mod 7, we have

$$3^6 6! \equiv 6! \mod 7.$$

6! has a multiplicative inverse mod 7, since its a product of numbers with multiplicative inverses. Multiplying both sides of the equation by this number, we deduce a special case of Little Fermat, for $a = 3, p = 7$:

$$3^6 \equiv 1 \mod 7.$$

Theorem 2. Let $N = p$ be a prime. Consider the power function $f(x) = x^e \mod p$ with domain equal to the residue numbers for p. Let e be chosen relatively prime to $p - 1$, $gcd(e, p - 1) = 1$. Let d be a multiplicative inverse of e mod $p - 1$ $(0 < e, d < p - 1)$. Then $g(x) = x^d \mod p$ is the inverse function of f.

Proof. Notice that if $x = 0$ the result holds. Thus we can assume $x = a$, a non-zero residue number for p. Since e and d are multiplicative inverses mod $p - 1$, we have

$$ed = 1 + m(p - 1)$$

for some counting number m. Thus

$$g(f(a)) \equiv g(a^e) \equiv (a^e)^d \equiv a^{ed}$$

$$\equiv a^{1+m(p-1)} \equiv a^1(a^{p-1})^m \equiv a(1^m) \equiv a \mod p$$

by Fermat's Little Theorem! This shows that g is the inverse function to f. $\square$

Example 6. For $p = 11$ and $e = 3$, find d using this Corollary. Does your answer agree with the earlier example, where we found acceptable d by brute force?

We can use the Little Fermat Theorem to understand power encryption/decryption when the modulus is a product of two different primes, and this is the basis of RSA cryptography. You will be able to understand the proof below, but it may seem like it was pulled out of a hat. Actually, it was pulled out of Wikipedia - a great source for understanding new math concepts. In an actual semester-long course on number theory, this result would be understood in a broader context, and would seem more natural.

Theorem 3. (RSA decryption, when $N = pq$) Let $N = pq$ be a product of two distinct prime numbers. Define $N_2 := (p-1)(q-1)$. Let e be relatively prime to N_2. Then $f(x) = x^e \mod N$ has inverse function $g(x) = x^d \mod N$, where the domain and range for f and g are the residue numbers for N, and where d is the multiplicative inverse of e, mod N_2.

Proof. If the hypotheses of the Theorem hold, then the claimed encryption and decryption powers e, d are related by

$$ed = 1 + m(p-1)(q-1)$$

for some counting number m. If we can show that

$$x^{ed} \equiv x \mod N$$

for all of $N's$ residue numbers, then it will follow that the modular power functions $f(x), g(x)$ are inverses of each other.

The trick is to show $x^{ed} - x$ is a multiple of p and also a multiple of q. Since p and q are different primes, the prime factorization of $x^{ed} - x$ must then include a factor of $pq = N$, so that $x^{ed} - x \equiv 0 \mod N$ as desired.

We show $x^{ed} - x$ is a multiple of p: If x is a multiple of p then this is automatic. Otherwise, $gcd(x, p) = 1$, and the residue of $x \mod p$ is a non-zero number a. By Little Fermat,

$$x^{p-1} \equiv a^{p-1} \equiv 1 \mod p.$$

Thus

$$x^{ed} = x^{1+m(p-1)(q-1)} = x^1(x^{p-1})^{m(q-1)} \equiv x(1)^{m(q-1)} \equiv x \mod p.$$

Thus in both cases, $x^{ed} - x$ is a multiple of p. By repeating the argument above and interchanging the roles of p and q, we deduce that $x^{ed} - x$ is also a multiple of q. Thus $x^{ed} - x$ is a multiple of pq, since p and q are prime and have no common factors. In other words, $x^{ed} \equiv x \mod N$ as claimed. $\square$

Example 7. For $p = 3$ and $q = 5$ we have $N = 15$, $N_2 = 8$. For the encryption power $e = 3$, compare the decryption power d guaranteed by this corollary to the power(s) we found earlier from the mod 15 power table.

Exercise 4. Let $p = 23, q = 41$, so that $N = 943$. Pick encryption power $e = 7$. Find the auxillary modulus N_2 and a decryption power d. Notice that you are completing the details of step 7 in the Davis notes example, section 9. We will go through this example carefully in the computer lab on Friday, using MAPLE.

Remark 2. Now is a good time to explain how and why the RSA system works as a public key system. A good source for more details (and links to primary references) is the Wikipedia page for RSA. These ACCESS class notes still need a good bibliography - sorry!

Suppose Bob wants to send a message to Alice. Alice creates her public key as follows. She uses the computer to find two random distinct primes p and q, since it turns out there are quick and effective algorithms to test whether large random numbers are prime. In practice p and q are typically between 512 and 1024 bits long, i.e. those are the ranges for the number of digits in the binary representations. Converting to decimal, that means that p and q have between 155 and 308 base 10 digits, so that the modulus N has about twice as many. Alice keeps p and q secret. Using her secret auxillary modulus $N_2 = (p-1)(q-1)$ she picks a large encryption power e relatively prime to N_2, along with her secret decryption power d - computerized algorithms using the Euclidean algorithm and other number theory algorithms can do this quite quickly.

Alice's public key consists of the modulus N and her encryption power e, so that the public encryption function for residue numbers x is $E(x) = x^e \mod N$. This is believed to be a good public key system because it is believed that no one can figure out her decryption power d just knowing N, e. The reason this is believed to be true is that there is no known algorithm for factoring large composite numbers which is fast enough to recover p and q from sufficiently large N, at least in any reasonable time period like the age of the universe. There are slow algorithms: you could test every integer less than or equal to $\sqrt(N)$ as possible factors (since if N factors into two integers, the smaller one will be at most this large). But if N is at least 10^{310} this on requires on the order 10^{155} checks to find the smaller factor. Since the age of the universe is currently thought to be about $4(10^{17})$ seconds, this would require at least 10^{137} checks per second which is orders of magnitude beyond the capabilities of all the computers on earth working together. (One megaherz is 10^6 cycles per second.)

There are more efficient factorization algorithms, but none that are (publically) known that could do this sort of factorization in any reasonable time frame. According to Wikipedia, the largest RSA modulus N with secret p, q that has been successfully factored in the public world was about 10^{231}. Of course, just as the Euclidean algorithm works exponentially faster than trial and error to find multiplicative inverses in modular arithmetic, it could be that there is an as yet undiscovered exponentially quick algorithm for factoring large composite numbers.

There are many further subtleties involved with secure RSA systems, and as well it would not be unreasonable to assume that the National Security Agency has arranged things so that most commercial RSA systems are restricted in some way so that they can be broken with sophisticated techniques; NSA is powerful, has the best computers on earth, and likely hires more mathematicians (including some Utah graduate students and postdocs) than any other entity on earth.