Exponential and RSA Ciphers

In the section on Fermat's Little theorem, we proved the following, which we shall put to use:

Fermat's little theorem. If p is a prime number and a is any number not divisible by p, then

$$a^{p-1} \equiv 1 \bmod p$$

Fermat's theorem (Two Prime version). If p and q are different primes and a is any number not divisible by p or by q, then

$$a^{(p-1)(q-1)} \equiv 1 \bmod pq$$

Some examples:

$$5^{22} \equiv 1 \bmod 23,\ 7^{88} \equiv 1 \bmod 89,\ 2331^{6336} \equiv 6499$$

The first two depend on your knowledge that 23 and 89 are primes. The second depends on the factorization $6499 = 67 \cdot 97 = pq$. In this case $(p-1)(q-1) = 66 \cdot 98 = 6336$. Here 2332 is not divisible by 67 or by 97. Less obvious are

$$5135^{80908} \equiv 1 \bmod 81493 \text{ and } 31891^{548238} \equiv 1 \bmod 548239$$

The first congruence depends on the factorization $81493 = 227 \cdot 359$ into primes. The exponent 80908 is the product of 226 and 358 (each one less than the primes in the factorization of the modulus.) The second congruence depends on the fact that 548238 is a prime (gotten from the PrimeWizard program in the lab.)

Large numbers like the last examples are clearly difficult to handle manually, though computers can deal with them by factoring into primes and by determining if a number is prime. Such congruences are used in cryptography. However the prime numbers will have 100 to 200 digits! In this case, the above numbers will seem like chicken feed. And for such huge primes, computers today can determine a factorization into primes only with great difficulty. A fast computer might take 20 or so years working full time to determine if a number is prime. And factoring tremendous numbers is even more hopeless. So if a cipher scheme can be found which involves huge numbers and factoring them into primes, the code would be practically unbreakable even with the best computers and smartest minds.

We first show how Fermat's two result are used. We illustrate with $5^{22} \equiv 1 \bmod 23$. If we wish to find $5^{223} \bmod 23$, we write $223 = 22 \cdot 10 + 3$, so

$$5^{223} = 5^{22 \cdot 10 + 3} = (5^{22})^{10} 5^3 \equiv 1^{10} 5^3 = 5^3 = 125 \equiv 10 \bmod 23$$

This amounts to replacing the exponent 223 by its value mod 22. (We divided by 22 to get the remainder 3.) In general, to compute $a^s \bmod p$, where p and a are as before, we can replace s by its value mod $(p-1)$. Note: The congruence is mod p but the exponents of a can be taken mod $p-1$.

Important consequence of Fermat's little theorem. If p is a prime and a is a number not divisible by p, then $a^r \equiv a^s \bmod p$ when $r \equiv s \bmod (p-1)$.

Example. Find $17^{1388} \bmod 67$.
Answer: Working the exponent mod 66, we find $1388 \equiv 2 \bmod 66$. So

$$17^{1388} \equiv 17^2 = 189 \equiv 25 \bmod 67.$$

In exactly the same way, we can reduce a power mod n when n is the product of different primes p and q.

Important consequence of Fermat's little theorem (Two Prime Version). If p and q are different primes and a is a number not divisible by p or by q, then $a^r \equiv a^s \bmod p$ when $r \equiv s \bmod (p-1)(q-1)$.

Example. Find $7^{2763} \bmod 143$.
Answer: Note that $143 = 11 \cdot 13$. (If you didn't see this immediately, check 143 mod 11 by the alternating sum test.) So here $p = 11$ and $q = 13$, so $(p-1)(q-1) = 10 \cdot 12 = 120$. We can reduce the exponent 2763 mod 120. We have $2763 \equiv 3 \bmod 120$ so

$$7^{2763} \equiv 7^3 = 343 \equiv 57 \bmod 143.$$

Exponential Codes – One Prime. A code attempts to send a message which can be read only by someone who has the key to the code. The message will be a stream of letters, but this is usually coded and transmitted as a stream of numbers. The receiver then decodes these numbers to get the original stream and hence the letters and the message. For example, suppose we wish to sent the message

DONT COME BEFORE FIVE CLOCK

Ignoring spaces, we convert each of these letters into their number equivalent (alway using two digits and ignoring spaces) and we try to code the number stream

04151420031513050205061518050609220515031215031 1

We break this into chunks depending on our coding system – say chunks of 3 – and we then proceed to send coded version of the three digit numbers

041 514 200 315 130 502 050 615 180 506 092 205 150 312 150 311

A (one prime) exponential code is given by the formula $C \equiv P^e \bmod p$ where p is a fixed prime, and e is relatively prime to $(p-1)$. Here, P is the plaintext (as number) and C is its cipher. For example, let's take $p = 947$. Then $p - 1 = 946 = 2 \cdot 11 \cdot 43$. Let's take

$e = 53$ which is prime to 946 since it is not divisible by 2 or 3 or 503.[1] So in this case, the exponential cipher we are using will be

$$C \equiv P^{53} \bmod 947$$

Our "message" is 041 514 200 315 .... We code this one three digit number at a time. For $P = 041$, we get[2]

$$C \equiv 041^{53} \equiv 544 \bmod 947$$

Similarly, $P = 514$ is coded as $C \equiv 514^{53} \bmod 947$ giving $C = 390$, and the third triple digit number 200 is coded as $200^{53} \equiv 677 \bmod 947$. Proceeding in this way, we generate the cipher text for each part of our message and we send the message

$$544\ 390\ 677\ \ldots$$

confident that no one will understand. The receiver knows all about the prime 947 but she takes exponent 357. (This will be shortly explained.) So she decodes 544 to get $P = 544^{357} \bmod 947$. This works out to 41 or 041 since three digit number are understood. Similarly, $390^{357} \equiv 514 \bmod 947$ and $677^{357} \equiv 200 \bmod 947$. So the decoded message starts as 041 514 200 .... Making a stream of digits, we get 041514200...or 04 15 14 20 .... In letters, the decoded message is DONT....

Summarizing this technique, we chose $p = 947$ and $e = 53$ for coding three digit numbers, and we chose $d = 357$ for decoding:

$$C \equiv P^{53} \bmod 947;\ P \equiv C^{357} \bmod 947$$

The coder knows exponent 53, the decoder knows exponent 357, both know the prime 947.

What is the magic use of the exponents 357 and 53? How does that work? The answer is the 357 and 53 are *inverses* mod 946. (Note: 946 not 947!). It works because the coded message is $C \equiv P^{53} \bmod 947$. The decoder then take $C^{357} = P^{53 \cdot 357}$. But this exponent is congruent to 1 mod 946 since that's how 53 and 357 were chosen. Thus $C^{357} \equiv P^1 = P \bmod 947$. Note that we always get and code three digit numbers since that's the size of the remainders mod 947.

Summary of exponential coding mod p. For given prime p and number e relatively prime to $p-1$, choose d so that $ed \equiv 1 \bmod (p-1)$. Then the coding is given by $C \equiv P^e \bmod p$ and the decoding is given by $P \equiv C^d \bmod p$. In all cases P and C are chosen between 0 and $p-1$ inclusive, namely the possible remainders when dividing by p.

[1] The primes and the factorizations in this discussion were all done with the PrimeWizard software in the lab.

[2] This was all computed using the ModCalc software in the lab.

Exponential Codes – Two primes. The situation here is almost the same, but it is the basis for the RSA coding system, considered the most powerful, so far unbreakable coding system ever. We start with two different primes p and q. Take $n = pq$, and take e relatively prime to $(p-1)(q-1)$, and take d as an inverse of e mod $(p-1)(q-1)$. Then $ed \equiv 1 \bmod (p-1)(q-1)$. The coder is given the exponent e and the decoder the exponent d. Both know the number $n = pq$. The coder computes the code using the formula $C \equiv P^e \bmod n$ and the decoder decodes using the formula $P \equiv C^d \bmod n$. Everything on the surface is the same. For example, let's choose small numbers so we can transmit two digits (or one letter) at a time. Take $p = 11$ and $q = 17$. Then $n = pq = 187$. Here $(p-1)(q-1) = 160$ with prime divisors 2 and 5. Choose $e = 57$, say. Then we must find its inverse d mod 160. This calculates to (trust me) $d = 73$. The coder is given the number $n = 187$ and exponent $e = 57$. The decoder is given the number $n = 187$ and exponent $d = 73$. Say the coder wants to sent the message NO, or 14 15. He computes 14^{57} mod 187 which works out to be 20, and $15^{57} \equiv 49 \bmod 187$. the message is simply

20 49

The receiver gets the message 20 49. He decodes $20^{73} \equiv 14 \bmod 187$ and $49^{73} \equiv 15$. The decoded message is, naturally, 14 15 or NO!

Summary of exponential coding mod pq. For given primes p and q take $n = pq$ and number e relatively prime to $(p-1)(q-1)$, choose d so that $ed \equiv 1 \bmod (p-1)(q-1)$. Then the coding is given by $C \equiv P^e \bmod n$ and the decoding is given by $P \equiv C^d \bmod n$. In all cases P and C are chosen between 0 and $n-1$ inclusive, namely the possible remainders when dividing by n.

Public Key Cryptography. Imagine a company or a department with lots of people who have to send encrypted messages to one another. The person receiving the message is the only one who is able to decode the message and anyone can send an encrypted message to anyone else in the company. In order to do this, every person in the company is given a modulus number n and an exponent e. These numbers are listed in a directory for all to see, so anyone can send an encrypted message. (This makes the keys public.) But each person in the company is also given a decoding exponent d, and this is known only to her. So if I want to send a message to Mary, say, I look up her modulus n and her exponent e, and send the message. Mary receives the message and uses her decoding key d and decodes. Anyone else on the list can intercept the message but they can't decode - they don't have the decoding key d – they only have the modulus n and Mary's coding exponent e which is public. So Mary decodes and the others are frustrated.

The system is called the RSA system in honor of the inventors, R. Rivest, A. Shamir, and L. Adelman. It is based on the two prime system described above. Here $n = pq$ and e is relatively prime to $(p-1)(q-1)$. d is the inverse of e mod $(p-1)(q-1)$. Why, you may ask, is d such a secret decoder. Anyone knows John's modulus n and exponent e. So factor n as pq, and compute $p-1)(q-1)$. Then find d as the inverse of e mod $p-1)(q-1)$. So it would seem that all anyone needs is a little bit of congruence mathematics. But here's the

problem: p and q are tremendously large primes say 100 to 200 digits. Since $n = pq$, n has 200+ digits. How do you factor it. The fact is that this is a problem computers can do but the fastest computer, working full time, will take years[3] to accomplish the factorization. So far, after 25 or so years, this RSA system is regarded as unbreakable. Naturally, people are trying, and if they succeed, it's back to the drawing board!

[3] an estimated 4 million years for a 200 digit number!. And much more for a number with more digits.