

Fermat's Little Theorem

PIf p is prime and a is an integer with $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$

< For example: $4^{16} \equiv 1 \pmod{17}$

< And: $99^{16} \equiv 1 \pmod{17}$

< And: $1684187324^{16} \equiv 1 \pmod{17}$

< But: $34^{16} \equiv 0 \pmod{17}$

< How about: $7887^{995} \pmod{17}$?

Some Modular Arithmetic Rules

PIf $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then:

< $a + c \equiv b + d \pmod{m}$

< $a - c \equiv b - d \pmod{m}$

< $ac \equiv bd \pmod{m}$

PWe can iterate the multiplication property on the single congruence $a \equiv b \pmod{m}$ to obtain:

< $a^2 \equiv b^2 \pmod{m}$

< $a^3 \equiv b^3 \pmod{m}$

< $a^4 \equiv b^4 \pmod{m}$

< ...

< $a^n \equiv b^n \pmod{m}$

Using Modular Arithmetic Rules

PWhat it all means for us:

< When we wish to evaluate some expression \pmod{m} , we can replace quantities that are being

- added,

- subtracted,

- multiplied or

- raised to a power

< by any other quantity which is in the same congruence class, and the result will be the same

< However, we cannot change exponents in this fashion without changing the value of the expression

Evaluate: $(5762 \times 2874 + 123874^4) \pmod{5}$

$\equiv (2 \times 4 + 4^4) \pmod{5}$

$\equiv (8 + 6) \pmod{5}$

But Exponents are Different

PHow would we compute $7887^{995} \pmod{17}$?

< The base, 7887, we can replace with $7887 \pmod{17}$, which is 16, because

< $7887 \equiv 16 \pmod{17}$ so $7887^{995} \equiv 16^{995} \pmod{17}$

PBut we cannot similarly reduce the exponent $\pmod{17}$; we have no such rule. In fact, doing so would give us the wrong answer!

PBut Fermat's Little Theorem gives us a slick way to do it:

< FLT tells us that $16^{16} \equiv 1 \pmod{17}$

< Thus $(16^{16})^k \equiv 1 \pmod{17}$ for any positive integer k

< $995 = 62 \times 16 + 3$. So I'll write 16^{995} as $16^{62 \times 16 + 3}$, which is $16^{62 \times 16} \times 16^3 \equiv 1 \times 16^3$ which is just $16 \pmod{17}$.

RSA Encryption

P Alice wishes to be able to receive secure messages from Everybody

- < Alice obtains a key, K , and a decoder, D
- < Alice publishes her key, K , for Everybody to see
- < But she keeps D secret
- < Everybody can use K to encode the message
- < Only Alice can decode the message, using D

A Sample Usage of this Method

P Paul wants to purchase a book from Amazon.com

- < He goes to the Amazon.com site, selects a book, and then goes to the secure "check out" page
- < Amazon.com sends its public key, K , to Paul's browser
- < Paul enters his personal information, and clicks "send"
- < Paul's computer encrypts the information using K , and only Amazon.com can decrypt it

Another Sample Usage of RSA

P Mary wants to download her bank information

- < She goes to the bank site, which sends her its public key K
- < The computer generates a random key R
- < The computer encrypts R using K , and sends it to the bank
- < The bank decrypts the message to obtain R
- < Now only Mary and the bank know the key, and they can use it to securely send messages back and forth

The Mathematics of RSA

P **Fermat's Little Theorem:**

- < For prime and relatively prime to, we have $a^{p-1} \equiv 1 \pmod{p}$

P **The Euler N-function:**

- < For any positive integer n denotes the number of positive integers less than or equal to n which are relatively prime to n

The Mathematics of RSA

P Euler's Generalization:

< For any a and n , relatively prime to each other, $a^{N(n)} \equiv 1 \pmod{n}$

P Facts:

< For a prime p , $N(p) = p - 1$

< For primes p and q , $N(pq) = (p - 1)(q - 1)$

P Theorem:

< For any a and n , relatively prime to each other, there exists a b such that $ab \equiv 1 \pmod{n}$

The Mathematics of RSA

P Building the public and private keys:

< Find two large primes p and q , and let $n = pq$

< Select a public key, K , relatively prime to $(p - 1)(q - 1)$

< Find a private D such that $KD \equiv 1 \pmod{(p - 1)(q - 1)}$

< Select a portion M of the message with fewer than $\log_2(n)$ bits, so that M can be thought of as a number less than n

< M is encrypted by computing $M^K \pmod{n}$

< is decrypted by raising it to the D th power

< Iterate until the whole message is securely sent