# Notation

N denotes a positive integer

p denotes a prime

ZN = {0,1,2,...,N-1}

Can do addition and multiplication modulo N

Examples: let N = 12

9+8 = 5 in Z12

5 x 7 = 11 in Z12

5 - 7 = 10 in Z12

Arithmetic in ZN works as you expected, e.g. x·(y+z) = x·y + x·z in ZN

# Greatest common divisor

**Def**: For integers x,y: gcd(x,y) is the greatest common divisor of x,y

**Example**: gcd(12,18) = 6

**Fact**: for all integers x,y there exist integers a,b such that

a·x + b·y = gcd(x,y)

a,b can be found efficiently using the extended Euclid algorithm.

For the previous example:

gcd(12,18) = 6 = 2x12 -1x18 ⇒ a=2, b=-1

If gcd(x,y)=1 we say that x and y are relatively prime

# Modular inversion

Over the rationals, inverse of 2 is ½. What about ZN?

**Def**: The inverse of x in ZN is an element y in ZN such that x·y = 1 in ZN

y is denoted x-1.

Example: let N be an odd integer. The inverse of 2 in ZN is (N+1) / 2

Which elements have an inverse in ZN?

**Lemma**: x in ZN has an inverse if and only if gcd(x,N) = 1

**Proof**:

gcd(x,N)=1 ⇒ Exist a,b: a·x + b·N = 1 ⇒ a·x = 1 in ZN ⇒ x-1 = a in ZN

gcd(x,N)>1 ⇒ for all a: gcd(a·x, N) > 1 ⇒ a·x != 1 in ZN

gcd(x,N) = 2 ⇒ for all a: a·x is even ⇒ a·x != b·N+1

# Invertible elements

Def: Z\*N = (set of invertible elements in ZN) = { x in ZN: gcd(x,N) = 1 }

Examples:

1.- for prime p, Z\*p = Zp \ {0} = {1,2,...,p-1}

2.- Z\*12 = { 1, 5, 7, 11 }

For x in Z\*N, can find x-1 using extended Euclid algorithm.

# Solving modular linear equations

Solve: a·x + b = 0 in ZN

Solution: x = -b·a-1 in ZN

Find a-1 in ZN using extended Euclid. Run time: O(log2N)

# Fermat and Euler

## Fermat’s theorem

**Theorem**: Let p be a prime

**Example**:

p=5 34 = 81 = 1 in Z5

Fermat’s theorem **application**:

x in (Zp)\* ⇒ x · xp-2 = 1 ⇒ x-1 = xp-2 in Zp

another way to compute inverses, but less efficient than Euclid because the complexity of the exp operation is O(log3p)

Also, it only works if N in ZN is prime

## Application: generating random primes

Suppose we want to generate a large random prime

say, prime p of length 1024 bits (i.e. p = 21024)

Step 1: choose a random integer p in [ 21024, 21025 - 1]

Step 2: test if 2p-1 = 1 in Zp

If so, output p and stop. If not, goto step 1.

Simple algorithm (not the best). Pr[p not prime] < 2-60

The expected number of iterations of this algorithm is only a few hundreds.

There are much more efficient algorithms that generates primes without any doubt.

## The structure of (Zp)\*

**Theorem** (Euler): (Zp)\* is a cyclic group, that is

Exists g in (Zp)\* such that {1,g,g2,g3,...,gp-2} = (Zp)\*

g is called a **generator** of (Zp)\*

Example: p=7. {1,3,32,33,34,35} = {1,3,2,6,4,5} = (Z7)\*

Not every element is a generator: {1,2,22,23,24,25} = {1,2,4}

## Order

For g in (Zp)\* the set {1,g,g2,g3,...} is called the **group generated by g**, denoted <g>

**Def**: the order of g in (Zp)\* is the size of <g>

ordp(g) = |<g>| = (smallest a>0 s.t. ga = 1 in Zp)

Examples: ord7(3) = 6; ord7(2)=3 ; ord7(1)=1

**Theorem** (Lagrange):

For all g in (Zp)\* : ordp(g) divides p-1

## Euler’s generalization of Fermat (1736)

**Def**:

For an integer N define: (Euler’s function)

Examples:

;

For N=p·q:

**Theorem** (Euler):

Generalization of Fermat. Basis of RSA cryptosystem.

# Modular e’th roots

Let p be a prime and c in Zp

**Def**: x in Zp such that xe = c in Zp is called and **e’th root** of c

Examples:

71/3 = 6 in Z11 ⇒ 63 = 216 = 7 in Z11

31/2 = 5 in Z11 ⇒ 52 = 25 = 3 in Z11

11/3 = 1 in Z11

21/2 does not exists in Z11

## When does c1/e in Zp exist?

Suppose gcd(e,p-1) = 1

Then for all c in (Zp)\*: c1/e exists in Zp and is easy to find.

**Proof**:

let d = e-1 in Zp-1 then

d·e = 1 in Zp-1 ⇒ exists k in K s.t. d·e = k(p-1) + 1 ⇒

(cd)e = cde = ck(p-1) + 1 = [cp-1]k·c = [ 1 ]k · c = c in Zp

cp-1 = 1 because c lives in (Zp)\* and p is prime.

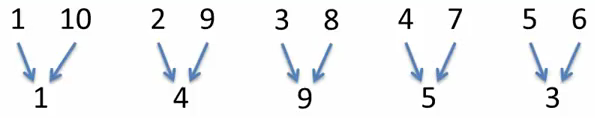
From the last part: c1/e = cd in Zp

## The case e=2: square roots

If p is an odd prime then gcd(2,p-1) != 1

**Fact**: in Zp\*, x→x2 is a 2-to-1 function

Example in Z11\*:



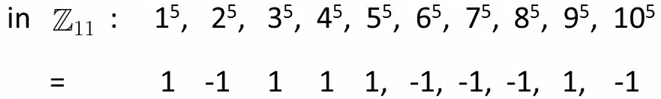
Def: x in Zp is a quadratic residue (Q.R.) if it has a square root in Zp

p odd prime ⇒ the number of Q.R. in Zp is (p-1)/2 + 1 ⇒ the 0 is always a Q.R.

## Euler’s theorem

**Theorem**: x in (Zp)\* is a Q.R. ⇐⇒ x(p-1)/2=1 in Zp (p odd prime)

Example:



Note: x!=0 ⇒ x(p-1)/2 = (x(p-1))½ = 11/2 in {1,-1} in Zp

It outputs 1 when Q.R. and -1 when not a Q.R.

**Def**: x(p-1)/2 is called the **Legrendre Symbol** of x over p (1798)

## Computing square roots mod p

Suppose p = 3 (mod 4)

**Lemma**: if c in (Zp)\* is Q.R. the sqrt(c) = c(p+1)/4 in Zp

**Proof**:

Lets prove that [c(p+1)/4]2 = c

[c(p+1)/4]2 = c(p+1)/2 = c(p-1)/2·c = 1·c = c in Zp

When p = 1 (mod 4), can also be done efficiently, but a bit harder. run time = O(log3 p)

Computing e’th roots mod N?

Let N be a composite number and e > 1

When does c1/e in ZN exists? Can we compute efficiently?

Answering these questions requires factorization of N (as far as we know)

# Arithmetic algorithms

Representing big nums

Representing an n-bit integer (e.g. n=2048) on a 64-bit machine ⇒ 32-bit blocks ⇒ n/32 blocks

Arithmetic

Given two n-bit integers