Today
- Finite fields
- Groups
- Number theory

Def: A field is a system $(S, +, \cdot)$ s.t.

- S is a set containing "0" & "1"
- $(S,+)$ is an abelian (commutative) group w. identity 0:

group laws:
$$\begin{cases} (a+b)+c = a+(b+c) & \forall a,b,c \in S \quad \text{(associative)} \\ a+0 = 0+a = 0 & \forall a \in S \quad \text{(identity 0)} \\ \forall a \in S \ \exists b \in S \text{ s.t. } a+b=0 & \text{(inverse)} \\ a+b = b+a & \forall a,b \in S \quad \text{(commutative)} \end{cases}$$

- $(S^*, \cdot)$ is an abelian (commutative) group w. identity 1:

$S^* = S \setminus \{0\}$

group laws:
$$\begin{cases} (a \cdot b) \cdot c = a \cdot (b \cdot c) & \forall a,b,c \in S^* \quad \text{(associative)} \\ a \cdot 1 = 1 \cdot a = a \\ \forall a \in S^* \ \exists b \in S^* \text{ s.t. } a \cdot b = 1 & \text{inverse} \\ a \cdot b = b \cdot a & \text{commutative} \end{cases}$$

Familiar fields: $\mathbb{R}$ (reals)

$\mathbb{C}$ (complex)

These are infinite fields (ie. fields w. infinitely many elements)

For crypto, we're usually working w. finite fields where $|S|$ is finite.

Ex: $(\mathbb{Z}_p, +, \cdot)$ where $+, \cdot$ are mod p

Thm: $\exists$ finite field w. q elements if and only if $q = p^k$ for some prime p and integer $k \geq 1$.

Moreover, for every such q there is a unique field consisting of q elements, denoted by $GF(q)$ (Galois Field)

$GF(p)$ for prime p is $(\mathbb{Z}_p, +, \cdot)$ where $+$ & $\cdot$ are mod p

$GF(p^k)$ consists of elements of the form $(a_0, a_1, \ldots, a_{k-1})$

where $a_i \in \mathbb{Z}_p$ $\forall i \in \{0, 1, \ldots, k-1\}$.

Think of each element as a poly of deg $< k$ over $\mathbb{Z}_p$

Mult. is multiplication as polynomials mod a fixed irreducible poly of deg k.

Common case: $GF(2^k)$

Note: $+$ and $\cdot$ can be performed efficiently

We will later demonstrate that inverse can be computed efficiently.

Next lecture : We will see how finite fields are used to share a secret (secret sharing scheme)

Often in cryptography we use finite groups (i.e set w. a single operation).

Common Groups : $\mathbb{Z}_p^*$, $\mathbb{Z}_n^*$, Q_p, Q_n, Elliptic curves (not today.)

$\mathbb{Z}_p^*$: multiplicative gp w. elements $\{1, \ldots, p-1\}$ and mult. mod p.
(p: prime)

$\mathbb{Z}_n^* = \{a \in \{1, \ldots, n-1\} \text{ s.t. } \gcd(a, n) = 1\}$.
(n: product of 2 primes, $n = p \cdot q$) (used in RSA)

Note : $|\mathbb{Z}_p^*| = p-1$

$|\mathbb{Z}_n^*| = n-p-q-1 = (p-1)(q-1) \triangleq \varphi(n)$. (Euler's function)

Def : The order of a group is the number of elements in the gp.

Sometimes, it is useful to have a prime order group

Note: If $p-1=2q$, where q is prime then

$Q_p = \{ a^2 : a \in \mathbb{Z}_p^* \}$ is a group of order q.

Def: If $p=2q+1$ then p is said to be a safe prime.

Q_p: Group of quadratic residues mod p.

$Q_n = \{ a^2 : a \in \mathbb{Z}_n^* \}$ = group of quadratic residues mod n.

Common operations: Exponentiation, inverse:

- Given a, b compute a^b
- Given a compute a^{-1}

How can this be done efficiently (i.e., in time poly(k)) (k = # of bits in a, b.)

Computing Exponentiation: Repeated squaring

To compute a, b compute a^b by

$$a^b = \begin{cases} 1 & \text{if } b=0 \\ a^{b/2} & \text{if } b>0 \text{ and even} \\ a \cdot a^{b-1} & \text{if } b \text{ odd} \end{cases}$$

Requires $\leq 2 \log b$ multiplications

$\approx O(k^3)$ time for k bit inputs

(A few miliseconds for 1024 bit integers).

Computing mult. inverses

Thm (Fermat's little thm):

$\forall$ prime p $\forall a \in \mathbb{Z}_p^*$ $a^{p-1} = 1$

Corollary $a^{-1} = a^{p-2}$ ← can be computed eff. by repeated squaring.

* Fermat's little thm is used to generate random k-bit prime numbers, as follows:

Choose at random $n \leftarrow \{0,1\}^k$, and choose a random $a \in \{1,\dots,n-1\}$

If $a^{n-1} = 1 \bmod n$ then choose n as the prime number.

O.w. try again.

Works because:

① Primes are dense: about $2^k / \ln(2^k)$ k-bit prime numbers

(Prime Number Theorem).

$\Rightarrow$ We expect to hit a prime after $\approx 0.69k$ tries.

② The test $a^{p-1} \stackrel{?}{=} 1 \bmod p$ works w.h.p. for random p

This test does not work for adv. chosen p.

- Miller-Rabin have a primality test that succeeds w.h.p. for every p
- [Agrawal-Kayal-Saxena 2002]: Gave a <u>deterministic</u> primality test.

Computing inverses in $\mathbb{Z}_n^*$

Thm (Euler): $\forall n \quad \forall a \in \mathbb{Z}_n^* \quad a^{\varphi(n)} = 1 \pmod n$

where $\varphi(n) = |\mathbb{Z}_n^*| = n - p - q + 1$

[More generally: **Lagrange Thm:** $\forall$ finite mult. gp G $\forall g \in G$, $g^{|G|} = 1$]

Corollary: $a^{-1} = a^{\varphi(n)-1} \bmod n$

* To compute a^{-1} this way requires knowledge of p & q.
* Often in crypto, we need to eff. compute inverses in $\mathbb{Z}_n^*$ without knowing the factorization of n.

Extended Euclid's Alg:

Euclids Alg Given $a, b \geq 0$ computes $\gcd(a,b)$.

$$\gcd(a,b) = \begin{cases} a & \text{if } b = 0 \\ \gcd(b, a \bmod b) & \text{o.w.} \end{cases}$$

Example: $\gcd(7,5) = \gcd(5,2) = \gcd(2,1)$

$= \gcd(1,0) = 1$.

- Running time is $\approx \log(a) \cdot \log(b)$ bit operations (polynomial runtime).

Thm: $\forall a, b \ \exists x, y$ s.t. $ax + by = \gcd(a,b)$.

Extended Euclids Alg: Given a, b outputs x, y s.t.

$ax + by = \gcd(a,b)$.

Example of this alg:

$7 = 7 \cdot 1 + 5 \cdot 0$
$5 = 7 \cdot 0 + 5 \cdot 1$
} initial values

$2 = 7 \cdot 1 + 5(-1)$ (subtract the two eqns)

$1 = 7(-2) + 5 \cdot 3$

x y.

Computing inverses w. Euclid's Extended Alg:

$a^{-1} \bmod n$:

Find x, y s.t. $ax + ny = \gcd(a,n) = 1$

$$a^{-1} = x \bmod n$$

So far: Groups $\mathbb{Z}_p^*$, $\mathbb{Z}_n^*$, Q_p^*, Q_n^* useful in crypto, and they can be generated efficiently and exponentiation & inverse can be found efficiently.

Often when we use $\mathbb{Z}_p^*$ we rely on the following assumption: For fixed $g \in \mathbb{Z}_p^*$ the function

$$f_g : x \longmapsto g^x \bmod p \quad \text{is one-way}$$

(i.e., hard to invert, and easy to compute which we have already established)

For which $g \in \mathbb{Z}_p^*$ is f_g OW?

Not for all g (eg. not for $g=1$)

" " $g=-1$

Can be OW only if the order of g is large as we define next.

Order of Elements & Generators:

Def: In any group G (eg. $\mathbb{Z}_p^*$, $\mathbb{Z}_n^*$), $\forall a \in G$

$$\text{Order}(a) = \text{least } u > 0 \text{ s.t. } a^u = 1 \quad (\text{in } G)$$

Recall Lagrange's Thm: $\forall$ finite gp G $\forall a \in G$

$$a^{|G|} = 1.$$

Corollary: $\forall a \in G$ $\quad \text{order}(a) \mid |G|$

(If $\text{order}(a) = u$ & $|G| = \alpha u + \beta$ $(\beta \in \{1, \ldots, u-1\})$,
then $1 = a^{|G|} = a^{\alpha u + \beta} = a^\beta$ - cont.)

Notation: $\langle a \rangle = \{a, a^2, \ldots, a^{\text{order}(a)}\}$

(subgroup generated by a.

Def: If $\langle a \rangle = G$ then a is a <u>generator</u> of G

Def: A finite group G is <u>cyclic</u> if $\exists$ generator $a \in G$.

Thm: $\mathbb{Z}_n^*$ is cyclic iff n is $2, 4, p^m$ or $2p^m$.

When we use group $\mathbb{Z}_p^*$, often we use it together w. a <u>generator</u> g, so that

$$f_g : x \longmapsto g^x$$ is a bijection from $\{0,1,\ldots,p-1\}$ to $\mathbb{Z}_p^*$.

$x \longmapsto g^x$ exponentiation

$g^x \longmapsto x$ discrete logarithm (DL)

- Computing DL is assumed to be hard in $\mathbb{Z}_p^*$ if g is a generator.
  Fastest known alg takes time $\geq 2^{\log p^{1/3}}$ ← sub-exp. alg.

Common public-key setup:

Public parameters:

p - large prime (eg 1024 bits).

g - generator of $\mathbb{Z}_p^*$

SK: random $x \in \{0,1,\ldots,p-1\}$

PK: $y = g^x \bmod p$.

Secrecy follows from DL assumption, which asserts

that the DL problem is hard.

Question: How do we find a generator of $\mathbb{Z}_p^*$?

Note: a random element in $\mathbb{Z}_p^*$ is not very likely to be a generator

Common solution: Choose safe prime $p = 2q+1$ (q prime)

& choose random $g \in \mathbb{Z}_p^*$ s.t. $g^2 \neq 1$ & $g^q \neq 1$.