CSCI 300

Number theory
+ crypto

Today

- Practice Final: wed.
- HW - due Friday
- email me for any solutions
- look for extra credit assignment due Monday

Algorithmic Number Theory

Increasingly, this area is of vital importance in computing.

Why?

- hash / passwords
- crypto

2 big things

- Math
- Engineering

Some definitions:

Most of these algorithms take place in a group or field.

What are these?

Group:

A set G which is equipped with an operation $*$ and a special element $e \in G$ s.t.

identity

① associative:

$$x * (y * z) = (x * y) * z$$

② $e * x = x \quad \forall x \in G$

③ $\forall x \in G$, there is $x' \in G$ s.t. $x * x' = e = x' * x$

inverse

Examples: $\mathbb{Z}, +$; $e = 0$

Is $\mathbb{Z}, *$? No! But $\mathbb{R}, *$ are

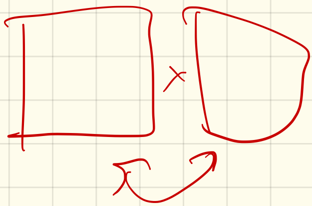
Abelian: A group where we also have **commutativity**:

$$x * y = y * x.$$

Ex: Ones on last slide

Non-example: matrices

$$A \circ B$$



Rings: These are sets R with **two** operations, $+$ and $*$ s.t.

1. R is abelian gp under $+$
2. $a*b = b*a$
3. $a*(b*c) = (a*b)*c$
4. Also have $1 \in R$ with $1 \neq e$ and $1*a = a$
5. $a(b+c) = ab+ac$ ← distributive

Ex: $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, + $\mathbb{C}$

Also: $\underbrace{\mathbb{Z}_m : \text{integers mod } m}_{\{0, \ldots, m-1\}}$

Let's back up a bit:

- Given a & b, $a \mid b$ means "a divides b".

In other words:

$$b = qa + r$$

for some $q \in \mathbb{Z}$ & $0 \le r \le a-1$

<u>Thm</u>: Consider $a, b, c \in \mathbb{Z}$.

- If $a \mid b$ and $b \mid c$, then $a \mid c$.
- If $a \mid b$ and $a \mid c$, then $a \mid (ib + jc)$ for <u>any</u> $i, j \in \mathbb{Z}$.
- If $a \mid b$ and $b \mid a$, then $a = b$

A number p is prime
$\iff 1, p$ are only divisors.
(so $d \mid p \Rightarrow d = 1$ or p)

If not prime, we say it is composite.

Fundamental Thm of Arithmetic:
Take $n > 1$, $n \in \mathbb{Z}$.
There are unique sets $\{p_1, \ldots, p_k\}$ and $\{e_1, \ldots, e_k\}$ such that

$$n = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$$

Greatest Common Divisors

gcd (a, b) = largest common divisor of a & b

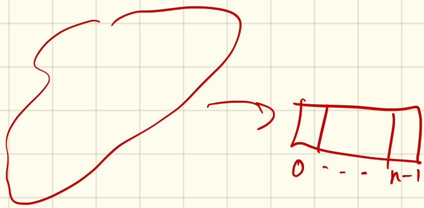
If $\gcd(a,b)=1$:

a & b are relatively prime

This is useful!

Relatively prime numbers are needed for:

- hash



GCD algorithm

Key lemma: Let a & b be 2 positive integers.
For any $r \in \mathbb{Z}$,
$$\gcd(a,b) = \gcd(b, a-rb)$$

Proof: Let $d = \gcd(a,b)$ &
$c = \gcd(b, a-rb)$

Goal: Show $c = d$. ✓

① $d \le c$: d divides a & b.
Use part 2 of earlier thm:
d divides: $1 \cdot b + (-r)b$
so $d \le c$ since d is a common divisor

② $c \le d$:
Know $c \mid b$ and $c \mid a-rb$
Consider: $\dfrac{a-rb}{c} = \dfrac{a}{c} - \dfrac{rb}{c}$ $\in \mathbb{Z}$
$\mathbb{Z}$?
so $c \mid a$ also. $\Rightarrow c \le d$.

Modulo:

if $r = a \bmod n$

$\Rightarrow r \in \{0, \dots, n-1\}$

Rewrite: $\exists q$ s.t.

$a = qn + r$

$\Rightarrow r = \underline{a - qn}$

Hrm... looks like key lemma!

remainders a good things to use!

Euclid's algorithm:

EuclidGCD(a, b):
    If b = a
        return a
    else
        EuclidGCD(b, a mod b)

Correctness:
    by lemma!

$a \bmod b = qn - a$

Runtime:

How many recursive calls?

Note: $\text{EuclidGCD}(a, b)$

$= \text{EuclidGCD}(b, a \bmod b)$

- 1^{st} input:

Let $a_i = 1^{st}$ input of i^{th} recursive call

so $a_{i+2} = a_i \bmod a_{i+1}$

Claim: $\forall i > 2,$
$$a_{i+2} < \frac{1}{2} \cdot a_i$$

pf:

If $a_{i+1} \leq \frac{1}{2} a_i$
then $a_{i+2} < a_{i+1}$
so done.

If $a_{i+1} > \frac{1}{2} a_i$
well, $a_{i+2} = a_i \bmod a_{i+1}$
$\Rightarrow a_{i+2} = a_i - a_{i+1} < \frac{1}{2} a_i$

Conclusion:

$$E(n) = 2 + E\left(\frac{n}{2}\right)$$
$$\Rightarrow O(\lg n)$$

Modular arithmetic is <u>key</u>:

Let $\mathbb{Z}_n = \{0, \ldots, n-1\}$

often called residues mod n

This is a common ring to work in:

- finite :
- has associativity, commutativity, identities, etc.

<u>Inverses:</u>

Sometimes!

Additive inverses:

Additive identity: in $\mathbb{Z}_m$, is 0

Do we always have an additive inverse?

Yes:

25 in $\mathbb{Z}_{128}$

$$25 + x = 128$$

$$x = 103$$

Multiplicative Inverses:

What is multiplicative identity?

1

Given $z \in \mathbb{Z}_n$, a multiplicative inverse z^{-1} is a number where:

Ex: 5 mod 9 ?

$5 \cdot x \bmod 9 \stackrel{?}{=} 1$

$x = 2$

Ex: 3 mod 9

NO: 3, 6, 9=0, 3, 6, 9=0, ...

Thm: An element $x \in \mathbb{Z}_n$ has an inverse in $\mathbb{Z}_n$

$\iff \gcd(x, n) = 1$

Side note: Why do we care?

① Why not use $\mathbb{R}$?

roundoff errors

② Why not use $\mathbb{Z}$?

Still infinite

③ How the heck does this matter in cryto?

Example: AES: Advanced Encrytion Standard

History:

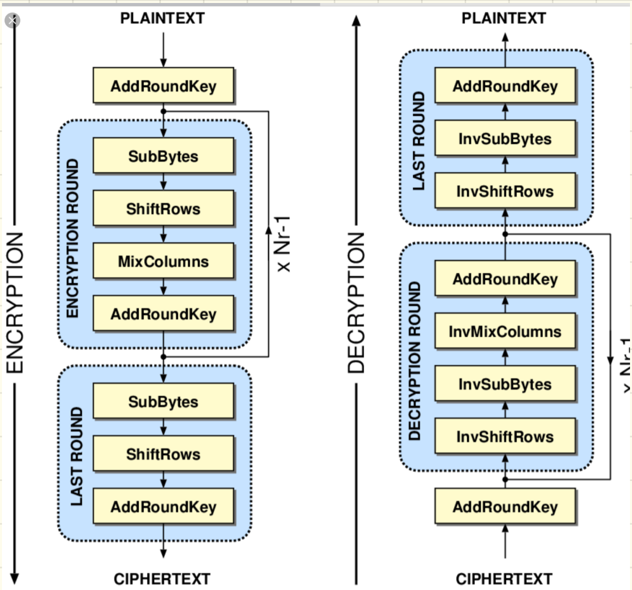
- In 1996, NIST issued a call to replace DES, prior encrytion standard
- In 1998, 15 algorithms were submitted.
- NIST spent years attacking & testing.
- The winner: Rijndael developed by 2 Belgian cryptographers.
- Officially approved in 2001.

How it works:

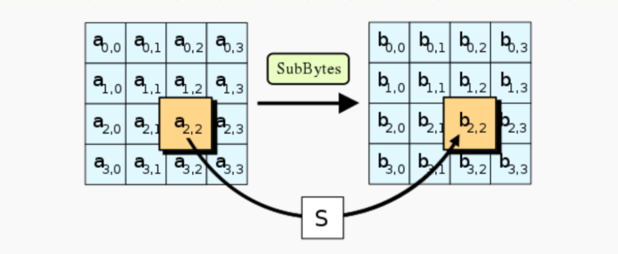
- Computation in $\mathbb{Z}_{256}$

Essentially, 4 operations:

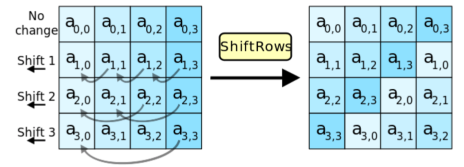
1. Substitute bytes.
2. Permute
3. Mix columns
4. Add round key
   (an XOR w/ portion of secret key)



1

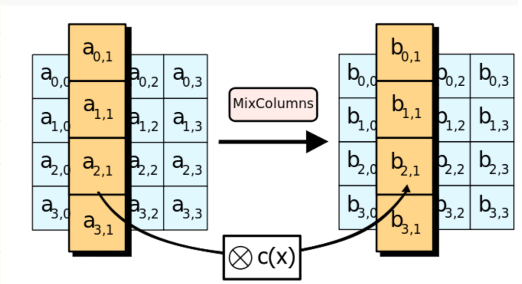


2



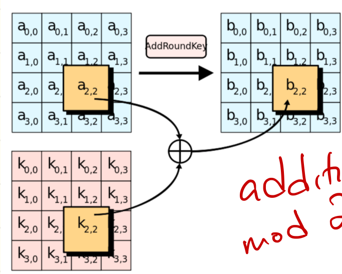
In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row.

3



In the MixColumns step, each column of the state is multiplied with a fixed polynomial $c(x)$.

4



addition mod 256

In the AddRoundKey step, each byte of the state is combined with a byte of the round subkey using the XOR operation (⊕).

AES (& all symmetric encryption) requires a secret, shared key.

Secure basically because you need to guess the secret key in order to attack.

Confusion: encrypting each bit requires more than 1 bit of key

Diffusion: Changing a bit doesn't only change 1 bit in output

Also - fast!

Basically linear time.

More interesting:

How do we agree on a secret key?

Best way:

However, impractical for things like web traffic or email.

Public Key Cryptography:

Use public information to send encrypted messages.

Diffie-Hellman Key exchange

From "New directions in cryptography"
by Diffie + Hellman in 1976

Daily conspiracy tidbit:
Actually discovered by UK government in 1973!

Key exchange

- Start with $\mathbb{Z}_p$
  (p prime or power of a prime)

These groups have multiplicative inverses:

i.e. $2x = 1 \mod 5$

The protocol: Alice + Bob:

- p + $s < p$ are both public
- Alice chooses secret $a < p$
  Bob chooses secret $b < b$
- Alice posts $A = s^a \bmod p$
  Bob posts $B = s^b \bmod p$

Alice computes:

Bob computes:

Example:

- $s = 2, \quad p = 29$
- Alice picks $a = 3$
  Bob picks $b = 7$

Why?

Common key is $k = s^{ab} \bmod p$

Public info: p, s, $A = s^a \bmod p$
and $B = s^b \bmod p$

What can an attacker try?

Hardness?

At its root, the key to why this is difficult is the discrete log problem:

Remember logarithms?

$\log_{10} 1000 =$

$\log_2 1024 =$

Here, discrete version:

Given A, find $\log_s A$

$= \log_s s^A$ modulo p

How hard?

This problem is connected to factoring

↳ NOT NP-Hard!

But no efficent algorithms are known.

Other key exchange algorithms work in other groups (like elliptic curves)

Next time: RSA + factoring