15-251

Great Theoretical Ideas in Computer Science

Clarifications of some of the homework Problems.

Problem 1

Problem 5

Problem 6

Raising numbers to powers, Cyrptography and RSA,

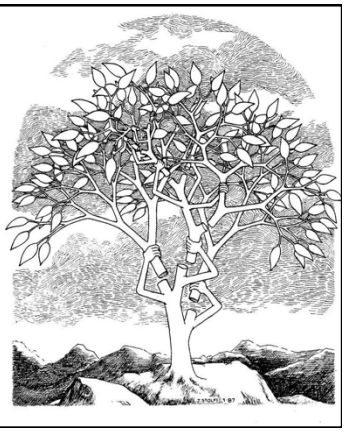
Lecture 14 (February 25, 2010)

$\equiv_p 1$

It's all done with splay trees.

QED.

Class Dismissed.



Not

How do you compute…

5^8 using few multiplications?

First idea:

$5 \quad 5^2 \quad 5^3 \quad 5^4 \quad 5^5 \quad 5^6 \quad 5^7 \quad 5^8$

$= 5*5 \quad 5^2*5$

How do you compute…

5^8

Better idea:

$5 \quad 5^2 \quad 5^4 \quad 5^8$

$= 5 * 5 \quad 5^2 * 5^2 \quad 5^4 * 5^4$

Used only 3 mults instead of 7 !!!

Repeated squaring calculates a^{2^k} in k multiply operations

compare with $(2^k - 1)$ multiply operations used by the naïve method

How do you compute…

5^{13}

Use repeated squaring again?

$5 \quad 5^2 \quad 5^4 \quad 5^8 \quad 5^{16}$ ✗

too high! what now?

assume no divisions allowed…

How do you compute…

5^{13}

Use repeated squaring again?

$5 \quad 5^2 \quad 5^4 \quad 5^8$

Note that $13 = 8+4+1$ $\quad 13_{10} = (1101)_2$

So $a^{13} = a^8 * a^4 * a^1$

Two more multiplies!

To compute a^m

Suppose $2^k \le m < 2^{k+1}$

$a \quad a^2 \quad a^4 \quad a^8 \quad \ldots \quad a^{2^k}$

This takes k multiplies

Now write m as a sum of distinct powers of 2

say, $m = 2^k + 2^{i_1} + 2^{i_2} \ldots + 2^{i_t}$

$a^m = a^{2^k} * a^{2^{i_1}} * \ldots * a^{2^{i_t}}$

at most k more multiplies

Hence, we can compute a^m while performing at most $2 \lfloor \log_2 m \rfloor$ multiplies

How do you compute…

$5^{13} \pmod{11}$

First idea: Compute 5^{13} using 5 multiplies

$5 \quad 5^2 \quad 5^4 \quad 5^8 \quad 5^{12} \quad 5^{13} \quad = 1\,220\,703\,125$

$= 5^8 * 5^{12} * 5$

then take the answer mod 11

$1220703125 \pmod{11} = 4$

How do you compute…

$5^{13} \pmod{11}$

Better idea: keep reducing the answer mod 11

5	5^2	5^4	5^8	5^{12}	5^{13}
	25		81	36	15
	3	9	4	3	4

(all mod 11)

Hence, we can compute $a^m \pmod{n}$ while performing at most $2 \lfloor \log_2 m \rfloor$ multiplies

where each time we multiply together numbers with $\lfloor \log_2 n \rfloor + 1$ bits

How do you compute…

$5^{121242653} \pmod{11}$

The current best idea would still need about 54 calculations

answer = 4

Can we exponentiate any faster?

OK, need a little more number theory for this one…

First, recall…

$Z_n = \{0, 1, 2, \ldots, n-1\}$

$Z_n^* = \{x \in Z_n \mid GCD(x,n) = 1\}$

Fundamental lemmas mod n:

If $(x \equiv_n y)$ and $(a \equiv_n b)$. Then

1) $x + a \equiv_n y + b$
2) $x * a \equiv_n y * b$
3) $x - a \equiv_n y - b$
4) $cx \equiv_n cy \Rightarrow a \equiv_n b$ i.e., if c in Z_n^*

Euler Phi Function Á(n)

Á(n) = size of Z_n^*

p prime $\Rightarrow$ Á(p) = p-1

p, q distinct primes $\Rightarrow$ Á(pq) = (p-1)(q-1)

~~**Fundamental lemma of powers?**~~

If $(x \equiv_n y)$
Then $a^x \equiv_n a^y$?

NO!

$(2 \equiv_3 5)$, but it is not the case that: $2^2 \equiv_3 2^5$

(Correct) Fundamental lemma of powers.

If $a \in Z_n^*$ and $x \equiv_{Á(n)} y$ then $a^x \equiv_n a^y$

Equivalently,

for $a \in Z_n^*$, $a^x \equiv_n a^{x \bmod Á(n)}$

How do you compute…

$5^{121242653} \pmod{11}$

121242653 (mod 10) = 3

$5^3 \pmod{11} = 125 \bmod 11 = 4$

Why did we take mod 10?

for $a \in Z_n^*$, $a^x \equiv_n a^{x \bmod Á(n)}$

Hence, we can compute $a^m \pmod n$ while performing at most $2 \lfloor \log_2 Á(n) \rfloor$ multiplies

where each time we multiply together numbers with $\lfloor \log_2 n \rfloor + 1$ bits

$343281^{327847324}$ mod 39

Step 1: reduce the base mod 39

Step 2: reduce the exponent mod Á(39) = 24

NB: you should check that gcd(343280,39)=1 to use lemma of powers

Step 3: use repeated squaring to compute 3^4, taking mods at each step

(Correct) Fundamental lemma of powers.

If $a \in Z_n^*$ and $x \equiv_{Á(n)} y$ then $a^x \equiv_n a^y$

Equivalently,

for $a \in Z_n^*$, $a^x \equiv_n a^{x \bmod Á(n)}$

How do you prove the lemma for powers?

Use Euler's Theorem

For $a \in Z_n^*$, $a^{Á(n)} \equiv_n 1$

Corollary: Fermat's Little Theorem

For p prime, $a \in Z_p^* \Rightarrow a^{p-1} \equiv_p 1$

Proof of Euler's Theorem: for $a \in Z_n^*$, $a^{Á(n)} \equiv_n 1$

Define $aZ_n^* = \{a *_n x \mid x \in Z_n^*\}$ for $a \in Z_n^*$

By the cancellation property, $Z_n^* = aZ_n^*$

$\prod x \equiv_n \prod ax$ [as x ranges over Z_n^*]

$\prod x \equiv_n \prod x \ (a^{\text{size of } Zn^*})$ [Commutativity]

$1 =_n a^{\text{size of } Zn^*}$ [Cancellation]

$a^{Á(n)} =_n 1$

Please remember

Euler's Theorem

For $a \in Z_n^*$, $a^{Á(n)} \equiv_n 1$

Corollary: Fermat's Little Theorem

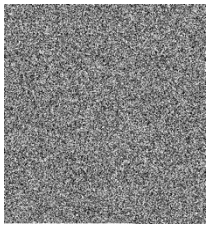
For p prime, $a \in Z_p^* \Rightarrow a^{p-1} \equiv_p 1$

Basic Cryptography

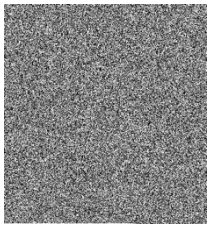
One Time Pads

The meeting
will be
in the town hall
at
midnight!

Come with your
parole officer!

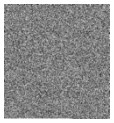


One Time Pads

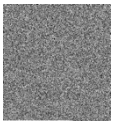


they give perfect security!

But reuse is bad



XOR =



Come with your
parole officer!

Can do other attacks as well

Agreeing on a secret

One time pads rely on having a shared secret!

Alice and Bob have never talked before
but they want to agree on a secret…

How can they do this?

A couple of small things

A value g in Z_n^* "generates" Z_n^* if
g, g^2, g^3, g^4, …, $g^{Á(n)}$
contains all elements of Z_n^*

Diffie-Hellman Key Exchange

Alice:
Picks prime p, and a generator g in Z_p^*
Picks random a in Z_p^*
Sends over p, g, g^a (mod p)

Bob:
Picks random b in Z_p^*, and sends over g^b (mod p)

Now both can compute g^{ab} (mod p)

What about Eve?

Alice:
Picks prime p, and a value g in Z_p^*
Picks random a in Z_p^*
Sends over p, g, g^a (mod p)

Bob:
Picks random b in Z_p^*, and sends over g^b (mod p)

Now both can compute g^{ab} (mod p)

If Eve's just listening in,
she sees p, g, g^a, g^b

It's believed that computing g^{ab} (mod p) from just this information is not easy…

also, discrete logarithms seem hard

Discrete-Log:
Given p, g, g^a (mod p), compute a

How fast can you do this?

If you can do discrete-logs fast,
you can solve the Diffie-Hellman problem fast.

How about the other way? If you can break the DH key exchange protocol, do discrete logs fast?

Diffie Hellman requires both parties to exchange information to share a secret

can we get rid of this assumption?

The RSA Cryptosystem

Our dramatis personae

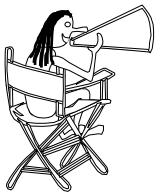

Rivest


Shamir


Adleman


Euler

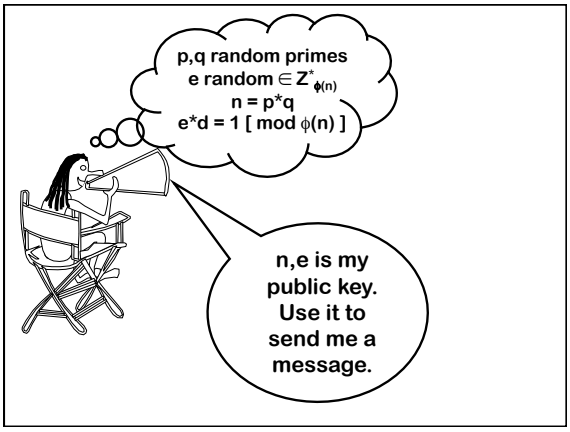

Fermat

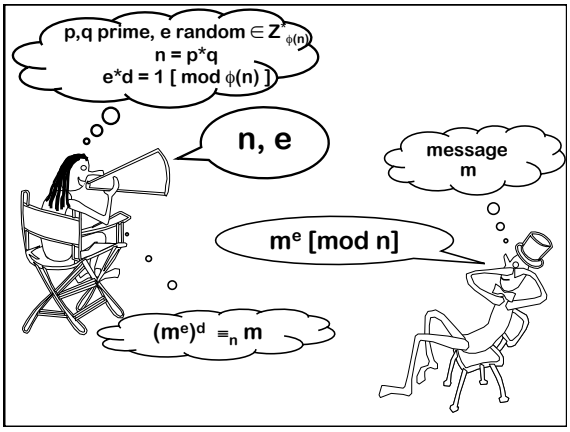


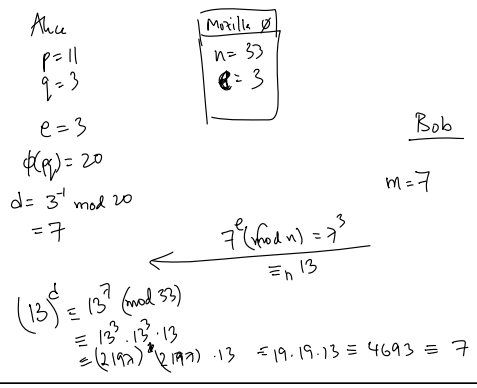
Pick secret, random large primes: p,q
Multiply n = p*q
"Publish": n

$\phi(n) = \phi(p)\,\phi(q) = (p-1)*(q-1)$
Pick random $e \in Z^*_{\phi(n)}$
"Publish": e

Compute d = inverse of e in $Z^*_{\phi(n)}$
Hence, $e*d = 1 \ [\text{mod } \phi(n)]$
"Private Key": d







Extended gcd on ocaml.
(egcd a b) returns a triple (x,y,g) such That g is the gcd(a,b) and g=ax+by.

```
let rec egcd a b =
  if a = 0 then (0,1,b) else
  let (x,y,g) = egcd (b mod a) a in
  (y-(b/a)*x, x, g)
```

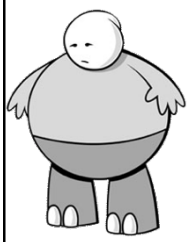
How hard is cracking RSA?

If we can factor products of two large primes, can we crack RSA?

If we know n and Á(n), can we crack RSA?

How about the other way? Does cracking RSA mean we must do one of these two?

We don't know (yet)…



Here's What You Need to Know…

Fast exponentiation

Fundamental lemma of powers
- Euler phi function $\phi(n) = |Z_n^*|$
- Euler's theorem
- Fermat's little theorem

Diffie-Hellman Key Exchange

RSA algorithm