Lecture 7
Spring 2020

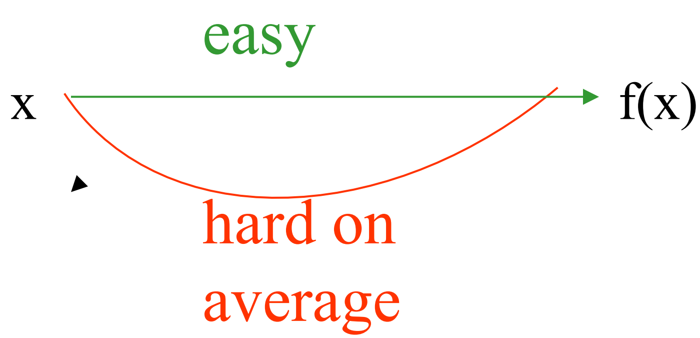
Shafi Goldwasser

Today:
Search for one-way functions

1. Discrete Log Problems in Cyclic Groups

2. Elliptic Logs over Elliptic Curves

Recall: One Way Function



Definition: f: $\{0,1\}^* \Rightarrow \{0,1\}^*$ is **a one-way function** if

1. **Easy to Evaluate:** ∃ PPT A s.t. A(x)=f(x)

2. Hard to Invert:

∀ PPT algorithm *Inverter*, ∀ sufficiently large n

Pr [x $\in \{0,1\}^n$:*Inverter*(f(x))=x' s.t. f(x)=f(x')]=negl(n)

Weak One-Way Function

Definition: $f: \{0,1\}^* \Rightarrow \{0,1\}^*$ is a **weak one-way** function

1. **Easy to Evaluate:** $\exists$ PPT algorithm A s.t. $A(x)=f(x)$

2. **Weakly Hard to Invert:** $\exists$ non-negligible ε
$\forall$PPT *Invertor*, $\forall$sufficiently large n
$\Pr[x \in \{0,1\}^n$: Invertor$(f(x)) \neq x'$ s.t. $f(x)=f(x'))$ $> \varepsilon(n)$

Note: we say "f has hard-core ε"
No ppt algorithm can succeed to invert for more than all but $\varepsilon(n)$ fraction.

Weak OWF iff Strong OWF

Amplification Theorem:
Weak one-way functions exist if and only if one-way functions exist

outline:
Say f is weak OWF with hard core e
Then $F(x_1 \ldots x_N)=f(x_1)|f(x_2)\ldots|f(x_N)$ for $N=2n/\varepsilon(n)$
is a one-way function $|x_i|=n$

There is a **HUGE blowup** in parameters going from n to n'=Nn
In practice, say if f is hard to invert on 1% on length 1000 inputs
Then F is hard to invert everywhere on 100,000,000 length inputs

We can do better with
concrete one way functions
Taking advantage of their algebraic structure

In Search of Concrete Examples of (weak) One-way functions

Review: Basic Group Theory

Basic Group Theory

Group $(G, \cdot)$ set with binary operation s.t.

- Closure: $\forall a,b \in G,\ a \cdot b \in G$
- Identity: $\exists\ 1 \in G$ s.t $\forall a,\ 1 \cdot a = a \cdot 1 = a$
- Inverse: $\forall a \in G,\ \exists\ a^{-1} \in G,\ a^{-1} \cdot a = 1$
- Associativity

Let G be a finite group

Order(G) = number of elements= $|G|$

Lemma: $\forall a \in G,\ a^{|G|} = 1$

Ex: $(Z_N,+)$ additive modulo N

Cyclic Groups

G is cyclic group if $\exists$ g $\in$G s.t. G={g, g^2, g^3,…, $g^{|G|}$}

Say that g is the generator of group G

Fact: Fix g generator for cyclic group G.

$\forall a \in G$, $\exists$ unique $1 \le i \le |G|$ s.t $a = g^i$

Say that i = discrete log of a w.r.t generator g

Computational Problems Associated with Cyclic Groups

- DLP in G: Given generator g and a $\in G$, compute $1 \le i \le |G|$ s.t. $a=g^i$ (the discrete log of a)

Looking for groups where

(1) group operation is easy

(2) DLP is hard

Number Theory

Elliptic Curves

Preliminaries: +, *, gcd

Let a,b >0 be n-bit integers.

Basic Terminology:

b|a (b divides a) if ∃ integer d >0 s.t. a=bd

gcd(a,b) = greatest integer d such that both d|a and d|b
e.g. gcd(9,21)=3

a and b are relatively prime if gcd(a,b)=1.

a is prime: has no divisors other than 1 or p

operation	Complexity
a+b	$O(n)$
ab	$O(n^2)$
gcd(a,b)	$O(n^2)$
a^b	$O(n^3)$

Easy ops asymptotically

In practice, when work with large integers, say n=160-4000 bits, use special `bignums' software

Modular Arithmetic

Let a, b, N> 0 be n-bit integers,

a mod N = remainder of a after dividing by N

e.g. 10 mod 3 =1, 7 mod 5=2

a=b mod N if (a mod N) = (b mod N)

b is the inverse of a mod N, denoted by a^{-1}

if a·b=1 mod N, e.g. 3^{-1} mod 7 = 5, (b exists if gcd(a,N)=1)

operation	complexity	
a mod N	$O(n^2)$	
a+b mod N	$O(n^2)$	
ab mod N	$O(n^2)$	
a^{-1} mod N	$O(n^2)$	[Euclid's algorithm]
a^b mod N	$O(n^3)$	[Repeated Doubling]

Algorithm to compute a^{-1} mod N

Let a^{-1} mod N= x s.t xa=1 mod N

Fact: x exists iff gcd (a,N) = 1

Euclid's algorithm: Given a,b integers.

Computes gcd(a,b) and x,y s.t. ax + by= gcd(a,b)

Main observation: if d|a and d|b then d|a-b

Poll: Can you use Euclid's algorithm to compute a^{-1} mod N ???

Algorithm to compute a^{-1} mod N

Let a^{-1} mod N= x s.t xa=1 mod N

Fact: x exists iff gcd (a,N) = 1

Euclid's algorithm: Given a,N.

Computes gcd(a,N)=1 and find x,y s.t. ax + Ny=1

Output x

Group Z_N^* ={1<=x<N s.t. (x,N) =1}

Theorem: Z_N^* is group under multiplication mod n

Proof: $\forall$a,b in Z_n^*, ab mod N in Z_N^* (closed)

1 in Z_N^* is the identity,

$\forall$a in Z_N^* , $\exists$ b s.t. ab=1 mod N

Euler Totient Function.

Order of Z_N^* = number of elements in Z_N^* = $\varphi(N)$

Theorem: $\varphi(p) = p-1$ for p prime,

$\varphi(N) = (p-1)(q-1)$ for $N=pq$, $\gcd(p,q)=1$

$\varphi(N) = \Pi_i p_i^{\alpha i-1}(p_i-1)$ for $N=\Pi p_i^{\alpha i}$

Theorem: $\forall$a in Z_N^* , $a^{\varphi(N)} = 1 \bmod N$

Examples

$Z_2^* = \{1\}$

$Z_3^* = \{1,2\}$

$Z_4^* = \{1,3\}$

$Z_5^* = \{1,2,3,4\}$

$Z_6^* = \{1,5\}$

$Z_7^* = \{1,2,3,4,5,6\}$

Observation: For prime p, $Z_p^* = \{1,2,\ldots,p-1\}$

Lets first focus on the the case of **p prime**

Group Z_p* for p prime

Theorem: If p is prime, then Z_p* is a cyclic group of order p-1

Ex: p=7, g=5 , Z_7* = {1,2,3,4,5,6} = {5,4,6,2,3,1}
= {5^i mod 7, i>0}

Let g be a generator of Z_p*, let a=g^b mod p

Call b the **discrete log** of a with respect to g

Useful Fact: if z = x+y mod (p-1) then $g^z = g^{x+y}$ mod p

Discrete Log Problem (DLP)

DLP: Given prime p, generator g of Z_p^*, a in Z_p^*,

find b such that $g^b = a \bmod p$

Notation: $DLP_{p,g}(a) = b$

Ex: p=7,g=5, the discrete log of 4 is 2 as $4=5^2 \bmod 7$.

Best Algorithm Known to Solve DLP

Runs in time $e^{O((\log p)^{1/3} (\log \log p)^{2/3})} \sim e^{O(n)^{1/3}}$ for n-bit primes p

Are there p,g for which DLP is known to be easy?

Not when p is prime

Furthermore Amplification: fix p, g:

can prove that if DLP is hard "at all", then its hard for all x.

Hardness somewhere ⇒ Hardness everywhere

Claim: Fix p prime, g generator.

If ∃ PPT algorithm B s.t. Prob [x in Z_p*: B(p, g, g^x) = x] > ε

Then ∃ probabilistic algorithm B' s.t. ∀ x, B'(p, g, g^x) = x

(B' runs in expected time polynomial in ε^{-1} and log p)

Proof idea:

B' (p,g,y)

1. Randomize: choose random 0< r<p-1;

   t=B(p,g, yg^r mod p)

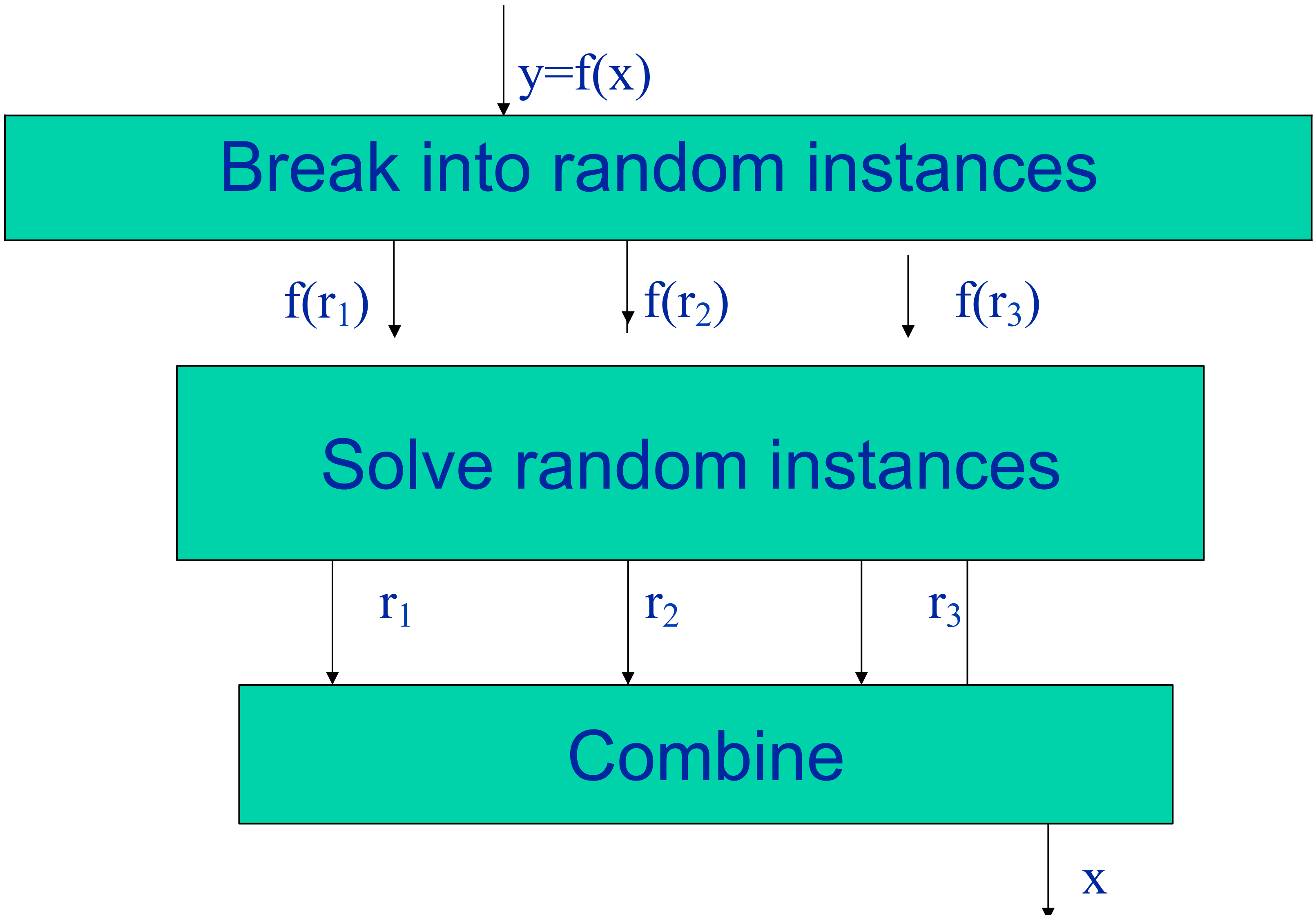
2. B succeeds⇒g^t=yg^r mod p ⇒ x =(t - r) mod (p-1)

   else repeat (go to step 1)

In expected 1/ε trials B will succeed

Corollary: If B' doesn't exist, neither does B. Namely, if $DLP_{p,g}$ is hard "at all" then $DLP_{p,g}$ (x) is hard for random x.

General : Random Self Reducibility



Corollary: If hard to invert for some f(x), hard to invert for random f(r)

Discrete Log ASSUMPTION (DLA)

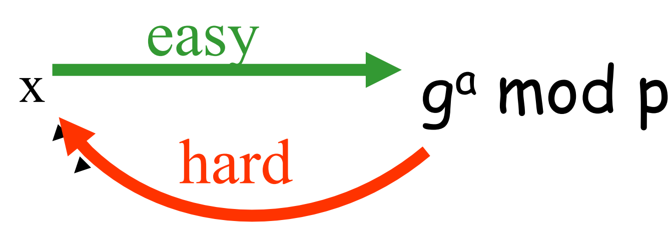
∀PPT algorithm A, suff. large n,

Prob (n-bit prime p, g generator for Z_p^*, $1 \leq b \leq p-1$:

$A(p,g,g^b) = b$) =negligible(n)

[Discuss: fixed prime, vs. random prime]

One Way Permutation CANDIDATE:



Modular Exponentiation

Let p prime, g be a generator for Z_p^*.

Define $EXP(p,g,b) = (p,g, g^b \bmod p)$

$EXP^{-1}(p,g,g^b \bmod p) = (p,g,b$ s.t. $1 \leq b \leq p-1)$

Discrete Log Problem(DLP)

✓ Example of One-Way Permutation

Example of OWF collection

Extra Structure: Specialized Applications

Collections of One-Way Functions

Definition: $F = \{f_i : D_i \to R_i\}_{i \in I}$ where I is a set of indices, and D_i , R_i are finite sets.

- **Sample a function**: ∃ PPT algo. $G(1^n)$ that selects f_i in F for i in $I \cap \{0,1\}^n$
- **Sample in Domain:** ∃ PPT algorithm S(i) that selects random x in D_i.
- **Easy to Evaluate**: ∃ PPT algorithm A s.t. $A(i,x) = f_i(x)$
- **Hard to Invert:** ∀ PPT Invert, ∀sufficiently large n, $Pr(i=G(1^n), x=S(i)$: Invert$(i,f_i(x))=x'$ s.t $f_i(x)=f_i(x')) <$ negligible(n)

OWF **Collection** Candidate: Modular Exponentiation

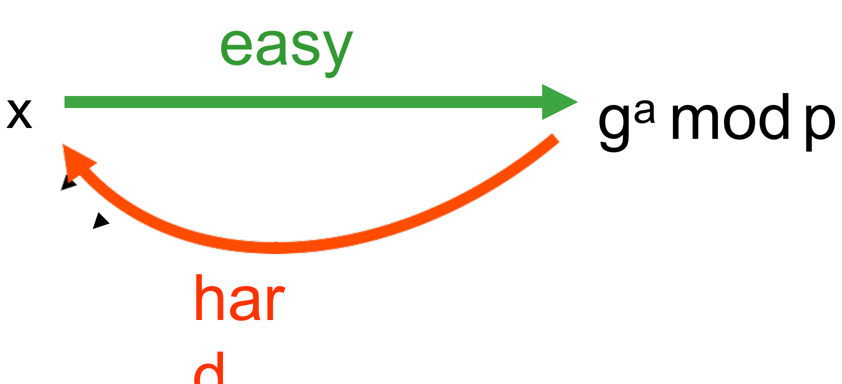
Let p prime, g be a generator for Z_p^*.

Define $EXP_{p,g}$:{1,...p-1}$\Rightarrow$ Z_p^*,

$EXP_{p,g}(a) = g^b \bmod p$

$EXP_{p,g}^{-1}(g^b \bmod p) = b$

$EXP = \{EXP_{p,g}\}_{p \text{ prime}, g \text{ generator}}$



Theorem: Under DLA, EXP is a collection of one-way functions.

$EXP = \{EXP_{p,g}\}_{p \text{ prime}, g \text{ generator}}$

Sample a function

- Need to generate a random prime p
- Need to generate a generator g

Easy to Evaluate: compute $EXP_{p,g}(x)$ in $O(n^3)$

Hard to Invert: By DLA

Generating Large Primes

Let $\pi(x)$ = number of primes < x

Prime Number Theorem:

$\lim \pi(x)/(x/\ln x) = 1$

Thus, about $1/(\ln x)$ numbers near x is prime.

By choosing at random numbers < x and testing for Primality, we will find a prime in $O(\ln x) = O(|x|)$ steps

Theorem [AKS 02]: Testing Primality is Easy.

For n-bit numbers,

- Current running time $O(n^6)$.
- Probabilistic algorithm: $O(n^4)$ time $/O(1/2^n)$ error.

Finding a Generator for Z_p^*

There are many generators for Z_p^* $O(1/\log n)$

- find a generator in $O(\log n)$ trials

How to check a given g is a generator?

Check that $g^{p-1}=1 \bmod p$,

$g^{(p-1)/qi} \neq 1 \bmod p \quad \forall \text{divisors } qi|(p-1)$

But do we know the factorization of(p-1)?

No.

Idea: Choose prime with p-1 in factored form -

Theorem: Under DLA, EXP is a collection of one-way functions.

Sample a function

Given security parameter n,

generate n-bit prime p and generator g for Z_p^* as follows:

Repeat

1. Generate a random number m in factored form $m = \Pi q_i^{\alpha i}$
2. let p-1=m. Test p for primality.

Until p is prime

Repeat

1. Choose random g in Z_p^*
2. Test if g is a generator for Z_p^* using factorization $(p-1)=\Pi q_i^{\alpha i}$

   Namely: if $g^{(p-1)/q} \neq 1 \bmod p \;\; \forall\, q|(p-1)$, g is generator

Until g generator

Special Interesting case: Strong Primes

- Restrict your prime to be a strong-prime p =2q+1 where q is a prime.

- In this case,
  - half the elements of Z_p* are generators
  - Can easily find and test a generator

- Most often used in practice

Discrete Log Problem(DLP)

- ✓ Example of One-Way Permutation
- ✓ Example of OWF collection

Extra Structure: Specialized Applications

Hard Problems to DLP

Computational Diffie-Hellman Problem (CDH):

given p,g, g^a mod p and g^b mod p,

compute g^{ab} mod p

Diffie Hellman Decisional Problem (DDH):

given g^a mod p, g^b mod p, and g^c mod p

distinguish c=ab mod (p-1) from

random 0<c<p-1

- Both problems are hard.
- Best solution known: first compute Discrete Log, same running time as Discrete Log.

Application 1:
Diffie Hellman Key Exchange

Let p be a prime,

g generator.

Party A chooses $1<x<p$ at random, set $y= g^x$,

and sends y to B over public channel

Party B chooses $1<z<p$ at random, set $w= g^z$,

and sends w to A over public channel

Joint **Secret Key** of A and B = g^{xz} =

w^x = [A can compute]

y^z [B can compute]

Security of Diffie-Hellman

- First key Exchange over public channels proposed
- Security
  - If CDH is hard adversary can't compute g^{xy} mod p
  - If DDH is hard adversary can't distinguish g^{xy} mod p from random

The hardness of DDH…later in class

Coin Flip over the Phone

A and B want to flip a coin over the telephone, but they don't trust each other

Idea 1: Alice flips a coin, tells Bob…BAD idea☹

Idea 2: Let p prime, g generator function

- A flips a coin c;
  - If c=0, A chooses even $0<x<p$
  - If c=1, A chooses odd $0<x<p$
  - Sends g^x mod p to B
- B guesses if x is even or odd
- A sends x to B. If guess is correct, then B wins, else A wins
- Is this a good idea?
- What is the bit security of x x from g^x mod p ?

The Quadratic Residues

$z \in Z_p^*$ is a quadratic residue mod p (square)
if $z=x^2$ mod p for some $x \in Z_p^*$;
and quadratic non-residue otherwise

Ex: p=7,

x mod p	1 2 3 4 5 6
x^2 mod p	1 4 2 2 4 1

squares ={1,2,4}
non-squares={3,5,6}

Let QR_p = quadratic residues mod p

Claim: QR_p is subgroup of Z_p^* of order (p-1)/2
Claim: Let g be a generator for Z_p^*
$y=g^i$ mod p, $0<i<p$ is a quadratic residue mod p
iff i is even

Decide if z is a quadratic residue mod p

Legendre Symbol of $z \in Z_p^*$ denoted $\left(\frac{z}{p}\right)$ = 1 if z is a quadratic residue mod p & -1 otherwise.

Claim[Easy to compute Legendre symbol]

$$\left(\frac{z}{p}\right) := z^{(p-1)/2} \bmod p$$

Proof: If $z = x^2 \bmod p$, then $z^{(p-1)/2} = x^{2(p-1)/2} = x^{(p-1)} = 1 \bmod p$.
z quadratic non-residue $\Rightarrow z^{(p-1)/2} = g^{(2i+1)(p-1)/2} = x^{i(p-1)+(p-1)/2} = g^{(p-1)/2}$.
Finally, g generator $\Rightarrow g^{(p-1)/2} = (g^{(p-1)})^{1/2} = (1)^{1/2} \bmod p = -1$ since it's one of the two (see below) roots of 1 and can't be 1.

Fact 2 : $y = x^2 \bmod p$ has 0 or 2 solutions when p is prime.
Proof: ∃solution x ⇒∃at least 2 solutions x & $-x = p-x \bmod p$.
Suppose ∃another $z \neq x, -x \bmod p$, $z^2 = x^2 \bmod p$ & $z^2 - x^2 = (z-x)(z+x) = 0 \bmod p$. Then, $p | (z-x)(z+x)$. As p is prime, it must divide either $(z-x)$ or $(z+x)$ ⇒ $z = x \bmod p$ or $z = -x \bmod p$. Contradiction

Bit Security of g^x mod p

Which information about x leaks from g^x mod p, $0<x<p$?

A: can compute LSB(x) from g^x mod p, by computing the Legendre symbol of g^x mod p,

Which information, if any, about x is well hidden by g^x mod p?

There must be some bit of x which is hard to compute, but which one?

Is there any bit of x which is **hard to predict** better than 50-50?

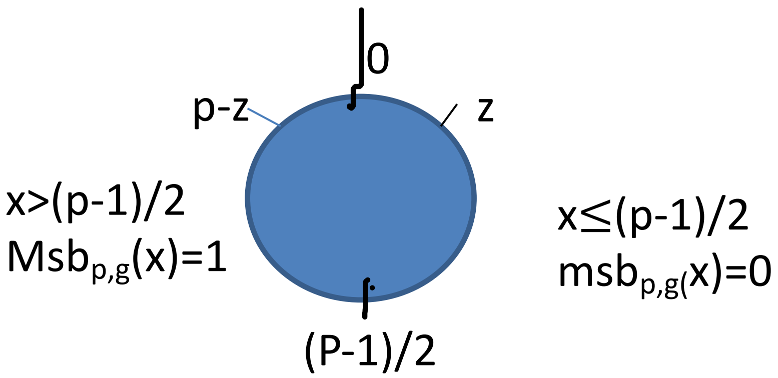
Theorem[MostSignificantBit is Hard Core Bit]:

Let $msb_{p,g}(x)$ = 0 if $x<(p-1)/2$ and 1 otherwise.

if ∃ PPT PRED, $c>0$ s.t.

$\text{Prob}[\text{PRED}(g^x \bmod p)=msb_{p,g}(x)] > ½+1/n^c$

then ∃ PPT that solves the discrete log problem mod p.



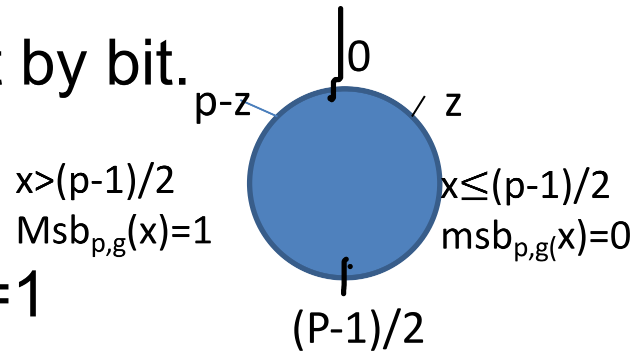
Proof Warm up: $y=g^x \bmod p$, $0<x<p$

,

Suppose $PRED(p,g,y)=MSB_{p,g}(x)$ for all y

$LSB(p,g,y) = 1$ if x is odd, 0 if x is even

IDEA: Will use LSB and the "oracle"
PRED for MSB to reconstruct $x= b_n \ldots b_1$ bit by bit.



<u>Discrete-Logarithm(p.g,y):</u>

0. Initialize $z := y \bmod p$ ($=g^x \bmod p$), $n=|p|$, $i=1$
1. Compute $b_i := LSB(p, g, z)$
2. If $b_i=0$, then $z=SQRT_p(z)$, else $z=SQRT_p(zg^{-1})$
3. If $PRED(p,g,z)=1$ then set $z=p-z$.
4. If $i< n$, let $i=i+1$, goto 1,
   else output $b_n \ldots b_1$

There are 2 square roots of g^{2i}
For g^i and $-g^{i/2}=g^{i/2}(-1)=g^i g^{(p-1)/2} = g^{i+(p-1)/2} \bmod p$
g^i is principal square root when $i <(p-1)/2$, otherwis

Proof Warm up 2: $y=g^x \bmod p$

Suppose ∀y: Pr [Pred(p,g,y)=$MSB_{p,g}$ (x)]>1-1/2n

Then, ∀y: Prob[DiscreteLogarithm (p,g,y) succeeds]=
Prob [Pred always succeeds]= $(1-1/2n)^n > 1/2$

<u>Algorithm Discrete-Logarithm'(p,g,y)</u>

Choose random 0<r<p ,
If Discrete-Logarithm(p, g, yg^r mod p) succeeds,
then x= Discrete-Logarithm(p, g, yg^r mod p) – r =x+r-r

Expected number of iterations =2

Summary: Hard vs. Easy

Z_p^* = {x < p and gcd(x,p) =1} for n-bit prime p

Let a,b in Z_{p^*}

operation	Complexity
a mod p	$O(n^2)$
a+b mod p	$O(n)$
ab mod p	$O(n^2)$
a^{-1} mod p	$O(n^2)$
a^b mod p	$O(n^3)$
Square or non-Square	$O(n^3)$
Solving Quadratic Equations mod p	$O(n^3)$
Lsb(x) from g^x mod p	

} easy

DL,DDH, DHP

MSB

} HARD?

What about other cyclic groups?

Elliptic Curve Cryptosystems

Elliptic Curves

Let a,b $\in F_p$ be s.t. $\gcd(4a^3+27b^2,p)=1$

An elliptic curve denoted as $E_{a,b}$ over finite field Z_p is the set of points (x,y) satisfying
$y^2=x^3+ax+b \bmod p$ PLUS a special identity point

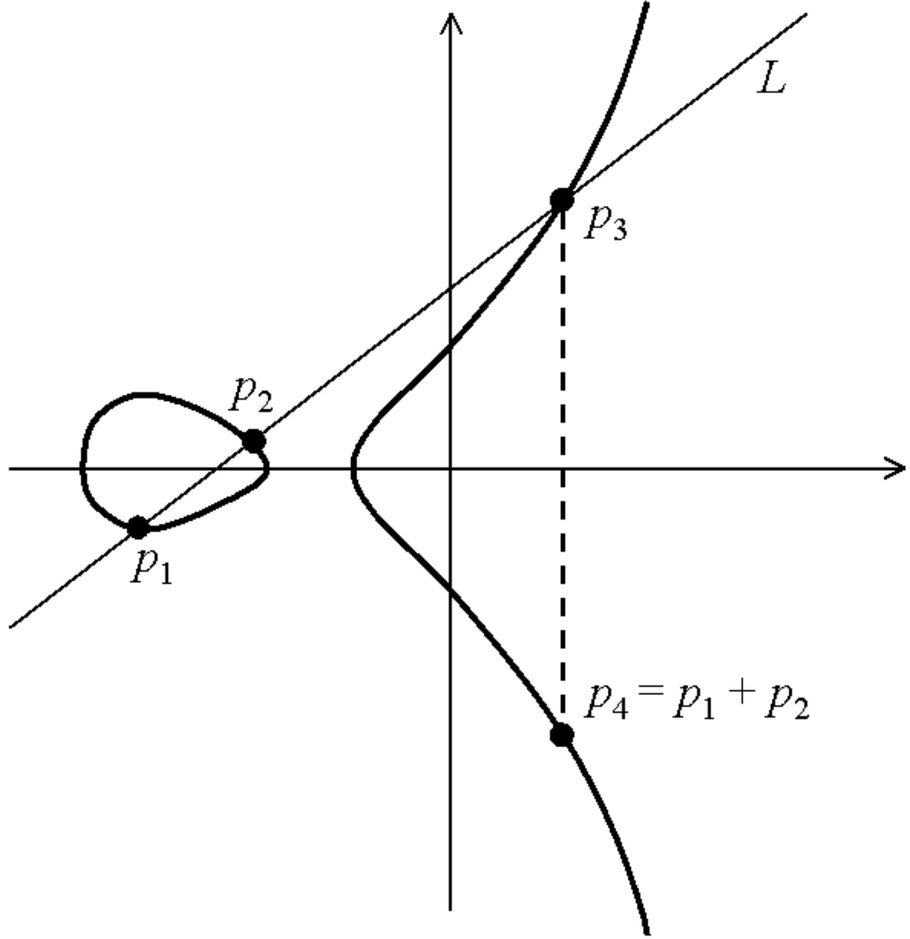
Under Addition of two points (see next slide) as group operation $E_{a,b}$ is a commutative group.

Elliptic Curve Discrete Log Problem (EDLP):

Given two points Q and P on the curve E,
find integer m s.t. Q = mP

Best Algorithm: exponential time $O(2^n)$ for general curve.

OWF candidate: f (m, P) = mP [Koblitz, Miller]



P1+ P2 = P4 where $s = (y_{P1} – y_{P2}) / (x_{P1} – x_{P2}) \bmod p$

$x_{P4} = s^2 – x_{P1} – x_{P2} \bmod p$ and $y_{P4} = -y_{P1} + s(x_{P1} – x_{P4}) \bmod p$

Why consider this group?

Elliptic Log problem(EDLP) may be harder than the discrete log problem(DLP)
Best algorithm known for EDLP is strictly exponential
(in contrast to DLP)

This means, we are able to use smaller groups with smaller security parameter (and operation cost) for same time invested to invert: an advantage for wireless devices w. low memory/ power

Can define ECDH & **E**DDH analogues over Elliptic Curves of CDH & DDH

ECDH seems hard,
but
EDDH problem is easy to decide.