Lecture 8:

Bit Security of DLP, Factoring, Squaring mod composites Trapdoor Functions and Permutations

Spring 2020

Shafi Goldwasser

Today

1. Bit Security of Modular Exponentiation , prime modulos g^x mod p

2. Elliptic Logs over Elliptic Curves

3. Trapdoor Functions

4. Z_n^*, composite n

The Quadratic Residues

$z \in Z_p^*$ is a quadratic residue mod p (square)

if $z=x^2$ mod p for some $x \in Z_p^*$;

otherwise, z is quadratic non-residue

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

if and only if i is even (i.e lsb(i)=0)

How to tell 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 exactly 2 solutions when p is prime.

Proof: ∃solution x ⇒∃at least 2 solutions x & $-x = p - x = xg^{(p-1)/2} \bmod p$.
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

There exists a PPT algorithm for solving $y=x^2 \bmod p$

Solve for x as follows.

Suppose eq. is solvable, then $z^{(p-1)/2} = 1 \bmod p$.

Case 1: $p=3 \bmod 4$, $(p-1)/2 = (4t+2)/2$

$z^{(2t+1)} = 1 \bmod p$

$(z^{(2t+1)})z = z \bmod p$

$(z^{(t+1)})^2 = z \bmod p$

output $x=z^{(t+1)} \bmod p$

Case 2: $p= 1 \bmod 4$, Harder, uses randomization, homework

Note: found both roots, x and $-x=p-x$.

For $x=g^i \bmod p$, $-x=g^i(-1)=g^i g^{(p-1)/2} = g^{i+(p-1)/2} \bmod p$

x is principal square root when $i < (p-1)/2$, otherwise $-x$ is

Bit Security of g^x mod p

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

A: can compute $lsb_{p,g}(x)$ from g^x mod p, by computing the Legendre symbol of g^x mod p. [$lsb_{p,g}(x)=0$ iff x is even iff g^x mod p is a quadratic residue]

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

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

Most Significant Bit (MSB)

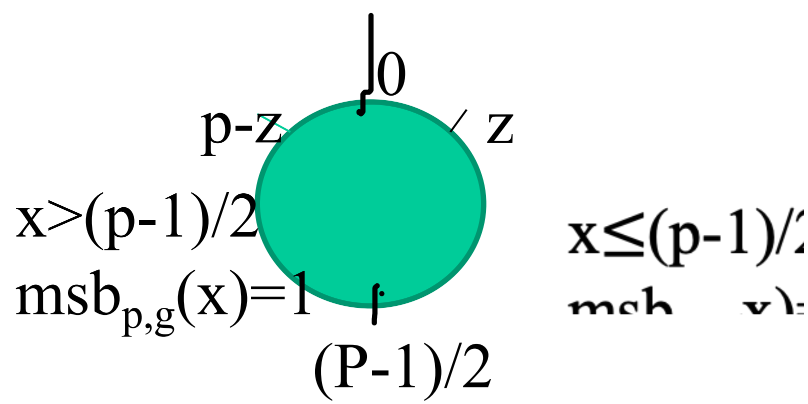
Theorem[MSB is Hard Core Bit]:

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

if $\exists$ PPT PRED, $c>0$ s.t.

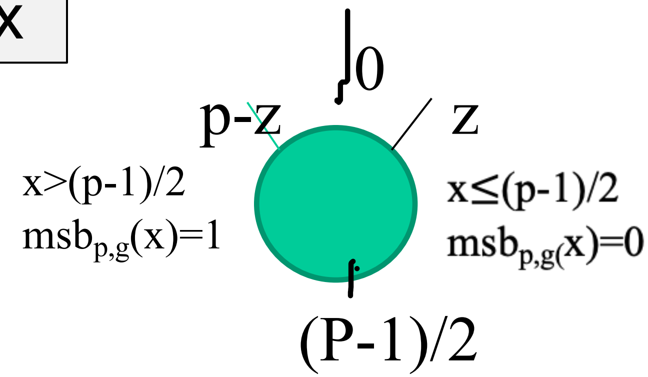
$$\text{Prob}[\text{PRED}(g^x \bmod p)=msb_{p,g}(x)] > ½+1/n^c$$

then can solve DLP in Z_p^*, p prime mod p by PPT algo.



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

Suppose $PRED(p,g,g^x)=msb_{p,g}(x)$ for all x



$lsb_{p,g}(y)$ =1 if x is odd, 0 if x is even

IDEA: Will use ability to compute lsb +

the “oracle” PRED for msb to reconstruct $x= b_n \ldots b_1$ bit by bit.

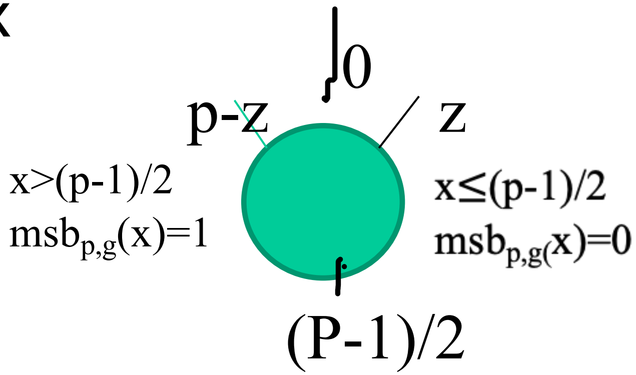
Discrete-Logarithm(p.g,y): Initialize $z:=y(=g^x \bmod p)$, $n=|p|$

Repeat from i=1 to n

1. Compute $b_i:=lsb_{p,g}(z)$ [e.g. $i=1, b_1=0,\ z=g^{b_n \ldots b_2 0} \bmod p$
   $i=1, b_1=1, z=g^{b_n \ldots b_2 1} \bmod p$]
2. If $b_i=0$, then $z=SQRT_p(z)$, else $z=SQRT_p(zg^{-1})$
   [But, there are 2 square roots:
   $SQRT(z)$ and $-SQRT(z)=SQRT(z)g^{(p-1)/2} \bmod p$. which one?]
3. If $PRED(p,g,z)=1$ then set $z=zg^{(p-1)/2} \bmod p$

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

Suppose $PRED(p,g,g^x)=msb_{p,g}(x)$ for all x



$lsb_{p,g}(y)$ =1 if x is odd, 0 if x is even

IDEA: Will use ability to compute lsb +

the "oracle" PRED for msb to reconstruct $x= b_n \ldots b_1$ bit by bit.

Discrete-Logarithm(p.g,y): Initialize $z:=y(=g^x \bmod p)$, $n=|p|$

Repeat from i=1 to n

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 $z=zg^{(p-1)/2} \bmod p$

output $x=b_n \ldots b_1$

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(p,g,) succeeds in computing $msb_{p,g}$

in every iteration of the algorithm]= $(1-1/2n)^n > 1/2$

<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

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 outcome… ☹
- Idea 2: Let p prime, g generator for Z_p^*
  - A flips a coin c;

    If c=0, A chooses even $0<x<p$

    If c=1, A chooses odd $0<x<p$

    A sends $g^x \bmod p$ to B
  - B guesses if x is $<(p-1)/2$ or $>(p-1)/2$
  - A sends x to B. If guess is correct, then B wins, else A wins

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	$O(n^3)$

easy

Solving Quadratic Equations mod p $O(n^3)$

Lsb(x) from g^x mod p

DLP,CDH, DDH

MSB

HARD?

Today

✓ 1. Bit Security of Modular Exponentiation , prime modulos

2. Elliptic Logs over Elliptic Curves

3. Trapdoor Functions

4. Z_n^*, composite n

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

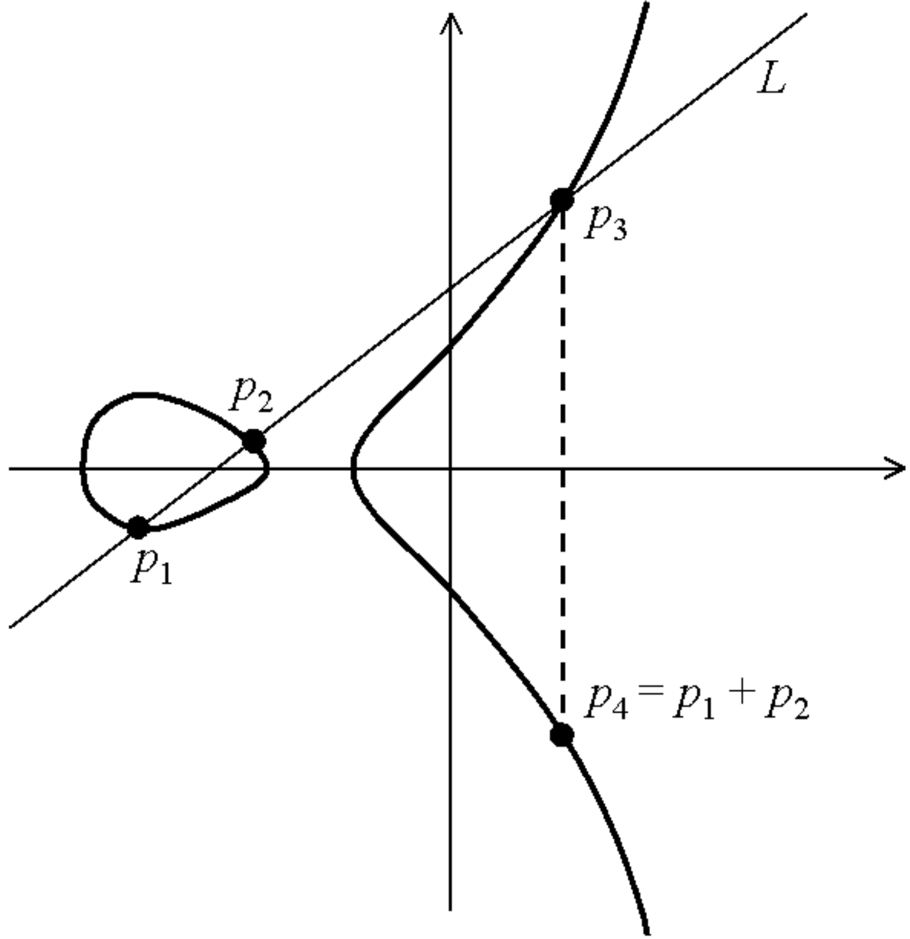
Under Addition of two points (see next slide) as group operation $E_{a,b}$ is a commutative group.

Elliptic Curve Discrete Log Problem (EDLP):

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

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

Today

✓ 1. Bit Security of Modular Exponentiation , prime modulos

✓ 2. Elliptic Logs over Elliptic Curves

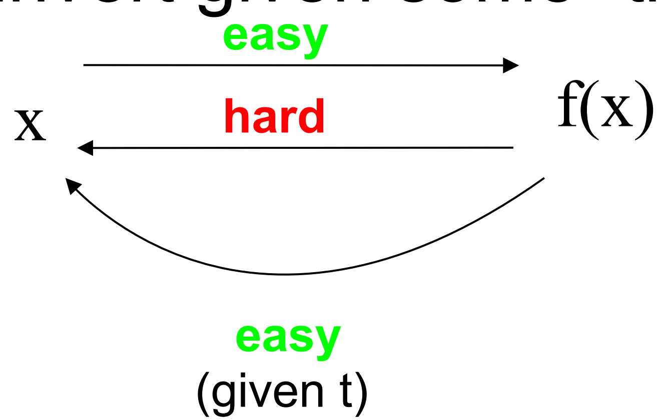
3. Trapdoor Functions

4. Z_n^*, composite n

Trapdoor Functions

Trapdoor Functions

- Informally: A trapdoor function family is a family of functions such that a randomly-selected function is:
  - Easy to compute
  - Hard to invert (given just f(x))
  - Easy to invert given some “trapdoor” t



Collections of Trapdoor Functions

Definition:

Let I be a set of indices, and D_i a finite set. A collection of trapdoor functions

is a collection of one-way functions

$$F = \{f_i : D_i \Rightarrow D_i\}_{i \in I}$$

- Generation: $\exists$ PPT algorithm G that on input security parameter 1^n selects a random $f_i \in F$ with $|i|=n$ with short trapdoor information t_i
- Trapdoor-ness: $\exists$ PPT algorithm INV,

  s.t $INV(i, f_i(x), t_i) = x'$ such that $f_i(x)=f_i(x')$

Today

- ✓ 1. Bit Security of Modular Exponentiation , prime modulos
- ✓ 2. Elliptic Logs over Elliptic Curves
- ✓ 3. Trapdoor Functions
- 4. Z_n^*, composite n

In Search of
Trapdoor Function Examples

Consider composite N

Composite N

Let N=pq where p,q are large primes.

Recall Z_N^* = $\{0<x< N \text{ s.t } \gcd(x,N)=1\}$
is a group under modular multiplication with
order $|Z_N^*| = \phi(N) = (p-1)(q-1)$.

EX: N=15,
$Z_{15}^*=\{1,2,4,7,8,11,13,14\}$
$\phi(15) = 8$

Note: Z_N^* may **not** be **cyclic** any more

Factoring

Factoring Algorithm:

Given N find divisor d s.t. d|N and 1<d<N

Best Known Algorithm: $e^{O((\log N)^{1/3} (\log \log N)^{2/3})}$

FACTORING ASSUMPTION:

∀PPT algorithms A,

Prob (A(N) outputs d|N s.t. d≠1,N) < neg(n)

n-bit N=pq, for |p|≈|q|

Squaring mod N Function[Rabin]

Let N=pq, p,q primes

Let $Rabin_N(x) = x^2 \bmod N$

$Rabin_N(x)$: $Z_N^* \longrightarrow QR_N$, QR_N = quadratic residues mod N

Properties of Squaring mod N

Let N=pq, p,q primes

Let $\text{Rabin}_N(x) = x^2 \bmod N$

$\text{Rabin}_N(x): Z_N^* \longrightarrow QR_N$, QR_N = quadratic residues mod N

Observations to be proven:

- Rabin_N is 4-1 function so not uniquely invertible
- **Trapdoor**: If factorization of N is known there exists a PPT algorithm for computing square roots mod N
- **Collection is One-Way if Factoring is hard**: If only N is known, computing square roots mod N is provably as hard as factoring.

To prove what we need,
Let us Digress

Effective Chinese Remainder Theorem (CRT)

Let N=pq be product of two distinct primes.

$\forall z \in Z_N$ map $z \rightarrow (z \bmod p, z \bmod q)$.

This mapping is a one-to-one and onto.

Furthermore, it is polynomial time to compute and invert.

Namely, given (z1,z2) where $z1 \in Z_p$ & $z2 \in Z_q$

can compute unique z in Z_N s.t

z=z1 mod p and z =z2 mod q

Chinese Remainder Theorem (CRT)

Proof: Let N=pq be product of two distinct primes.

Compute c_1 and c_2 s.t.

$c_1=1$ mod p and 0 mod q and

$c_2=1$ mod q and 0 mod p

How?

c_1: Compute b_1 s.t. $b_1q=1$ mod p and set $c_1=b_1q$, Check!

c_2: Compute b_2 s.t. $b_2p=1$ mod q and set $c_2=b_2p$. Check!

Call these the CRT coefficients

Given (z1,z2) where $z1 \in Z_p$ and $z2 \in Z_q$, set

$z=c_1z1+c_2z2$

Claim: Then z=z1 mod p and z=z2 mod q.

General Version: Chinese Remainder Theorem (CRT)

Let $p_1 \ldots. p_t$ s.t. $\gcd(p_i, p_j) = 1$ and $N = \Pi p_i$

and $x_1 \ldots. x_t$ be integers in Z_{p_i} respectively.

Then there is a unique solution x mod $N = \Pi p_i$

$x = x_1 \bmod p_1$

$x = x_2 \bmod p_2$

.

$x = x_n \bmod p_t$

and x can be easily computed from x_i's.

Example CRT

Given $p=3$, $q=7$, $z_1=2$, $z_2=5$, compute $z < 21$

Such that $z=z_1 \bmod p$ and $z=z_2 \bmod q$

Compute CRT coefficients

$c_1= 7$, since $7 \bmod 3 =1$, $7 \bmod 7 =0$ and

$c_2=15$, since $15 \bmod 3=0$, $15 \bmod 7 = 1$

Given c_1 and c_2, compute x as follows

$x =c_1z_1+ c_2z_2 = 2*7 + 5*15 = 89 \bmod 21 = 5 \bmod 21$

Use CRT to show z in QR_N if and only if z_1=z mod p in QR_p & z_2=z mod q in QR_q

⇐Say z_1 mod p in QR_p & z_2 mod q in QR_q

let x_1 s.t. $x_1^2 = z_1 \bmod p$

and x_2 s.t. $x_2^2 = z_2 \bmod q$

set $x = x_1 c_1 + x_2 c_2 \bmod N$

for $c_1 = 1 \bmod p$ and $0 \bmod q$

and $c_2 = 1 \bmod q$ and $0 \bmod p$

define $z = z_1 c_1 + z_2 c_2$

Claim: $z = x^2 \bmod N$, therefore z in QR_N

⇒If z in QR_N then $z = x^2 \bmod N$

implies $z = x^2 \bmod p$ (z mod p in QR_p)

and $z = x^2 \bmod q$ (i.e z mod q in QR_q)

Use CRT to show
z in QR_N if and only if
z_1=z mod p in QR_p & z_2=z mod q in QR_q

$\Leftarrow$Say z_1 mod p in QR_p & z_2 mod q in QR_q

let x_1 s.t. $x_1^2 = z_1$ mod p

and x_2 s.t. $x_2^2 = z_2$ mod q

set $x = x_1 c_1 + x_2 c_2$ mod N

for c_1 = 1 mod p and 0 mod q

and c_2 = 1 mod q and 0 mod p

define $z = z_1 c_1 + z_2 c_2$

Claim: $z=x^2$ mod N, therefore z in QR_N

$\Rightarrow$If z in QR_N then $z=x^2$ mod N

implies $z=x^2$ mod p (z mod p in QR_p)

and $z=x^2$ mod q (i.e z mod q in QR_q)

Finished Digression

Can now establish the necessary facts about the Rabin trapdoor function candidate

1. $\text{Rabin}_N(x)$ is 4-to-1 Function

Let $z=x^2 \bmod N$

Then $\exists\ x_1$ s.t. $x_1^2 = z \bmod p$ and
x_2 s.t. $x_2^2 = z \bmod q$

The following are the 4 distinct roots of z mod N:

$x = x_1 c_1 + x_2 c_2$ and $-x = N - x \bmod N$

$x' = -x_1 c_1 + x_2 c_2$ and $-x' = N - x' \bmod N$

for c_1 and c_2 CRT coefficients

Check !!!

2.Trapdoor: Given Factorization of N, Computing Square Roots mod N is easy

Let $N=pq$ and $z=x^2 \bmod N$.

$SQRT_N(p,q,z)$:

- Compute x_1 s.t. $x_1^2 = z \bmod p$
- Compute x_2 s.t. $x_2^2 = z \bmod q$
- Compute $c_1 = 1 \bmod p$ and $0 \bmod q$ (by CRT)
- Compute $c_2 = 1 \bmod q$ and $0 \bmod p$ (by CRT)
- Output $x = x_1 c_1 + x_2 c_2$

Recall: can compute square roots mod primes

3.Without trapdoor, Computing Square Roots mod N As Hard As Factoring N

Theorem: If ∃ PPT A s.t. $A(N,y)=x$ for $y=x^2$ mod N, then ∃ PPT algorithm to factor N.

Pf: 1. On input N, choose a random r in Z_N^*.

2. Compute $x=A(N,r^2 \bmod N)$.

3. If x = +/-r mod N [with prob ½], goto 1 [no use, already know it]

Otherwise $x^2 = r^2$ mod N but $x \neq r$ mod N and $x \neq -r$ mod N

[which implies either $x \neq r$ mod p or $x \neq r$ mod q]

4. Output gcd (N, x-r).

Claim: gcd(N, x-r) = p or q. **Pf:** Since $x^2 - r^2 = (x+r)(x-r) = 0$ mod N, but $x+r \neq 0$ mod N & $x-r \neq 0$ mod N, either $p|(x-r)$ or $q|(x-r)$ but not both, thus gcd(x-r,N) = p or q QED

3'.Squaring is hard to invert on the average as in the worst case

Theorem:

If ∃ PPT A s.t. Prob[A(N,y)=x s.t.y=x^2 mod N]> ε,
then ∃ PPT A' s.t. Prob[A(N)=d s.t d|N and d≠1,N]> $1-\delta$
and A' runs in time poly(ε^{-1}, δ^{-1}, log N)

Proof: Choose k s.t. $1/\varepsilon^k<\delta$

Repeat $2\varepsilon^{-1}k$ times

1.choose a random r in ZN*.

2. Compute x=A(N,r^2 mod N).

3. If x = +-r mod N (with prob 1/2), goto 1

   Otherwise $x^2 = r^2$ mod N but x≠r mod N &x ≠-r mod N

   [which implies either x ≠ r mod p or x ≠ r mod q]

4. Output gcd (N, x-r).

Prob[A' fails to factor N] $<$

Pr[an iteration fails]$^{\#\text{iterations}}$ $<$

$e^{-k} < \delta$

A Collection of Trapdoor Functions

Define Rabin = { $Rabin_N$ where N=pq, p,q primes s.t. |p|=|q|=n }

Theorem: Under Factoring-assumption, Rabin is a collection of trapdoor functions

(recall 1/n n-bit integers is prime)

Generation: Choose n-bit p,q and test for primality. If primes set N=pq, $trapdor_N$ = {p,q}

Evaluation: Computing $Rabin_N(x)$ takes $O(n^2)$ time

Hard to Invert: by Factoring Assumption

Trapdoor-ness: Given N, p and q can compute square roots mod N in $O(n^3)$

Associated Problem: Deciding Quadratic Residuosity modulo Composites

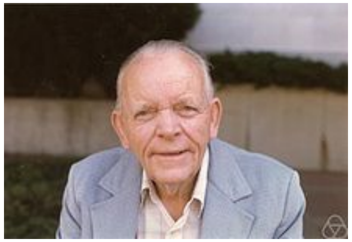
- Given factorization of N, easy to tell if z is quadratic residue
- Without factorization, don't know how to tell if z is a square mod N
- Jacobi Symbol = $\left(\frac{z}{N}\right) = \left(\frac{z}{p}\right)\left(\frac{z}{q}\right)$ an extension of the Legendre Symbol
  - easy to compute without the factorization of N, but
  - only gives partial information about if z is square

    (i.e if Jacobi symbol of z is -1 then z is definitely not square, but otherwise no information)

Quadratic Residuosity: Primes vs. Composites

Is $z=x^2$ mod N

•Lehmer: I am not a gambling man, wouldn't guess unless z is small (perfect squares)



Question: is it hard for a random $z \in Z_N^*$?

Quadratic Residuosity Assumption (QRA)

Let $QR_N(z) = \begin{cases} 0 \text{ if } z \text{ quadratic residue mod } N \\ 1 \text{ if } z \text{ is quadratic non-residue mod } N \end{cases}$

$$\left(\frac{z}{N}\right) = 1$$

Theorem (QR hard to predict if hard at all per n):

Let A be ppt s.t. $Prob_{(z/N)=1}[A(z,n) = QR_N(z)] > 1/2+\varepsilon$,
then $\exists$PPT B $\forall$ z in Z_n^* $Prob[B(z,n)=QR_N] > 1-\delta$

(B is Monte Carlo with runtime poly($1/\varepsilon, 1/\delta, |p|$)

- .

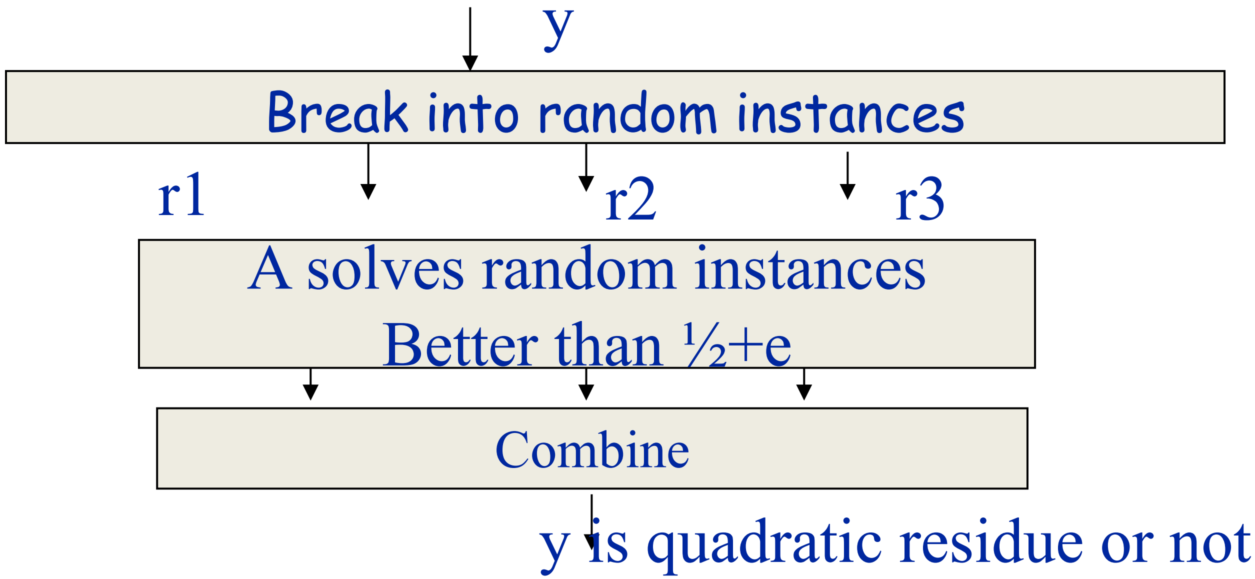
Quadratic Residues: Random Self Reducability

Theorem[GM]:.

If ∃ PPT A to decide quadratic residuosity with $\text{prob}_y > \frac{1}{2}+\varepsilon$ (over y's)

then ∃ PPT B to decide quadratic residuosity $\forall y \in Z_N^*$ w.p $>1-\delta$

A' runs poly(A, ε^{-1}, δ^{-1}) .



Corollary [Worst Case to Average]:

Fix n. QR is hard for *worst case* y⇒*its hard to for the average* y

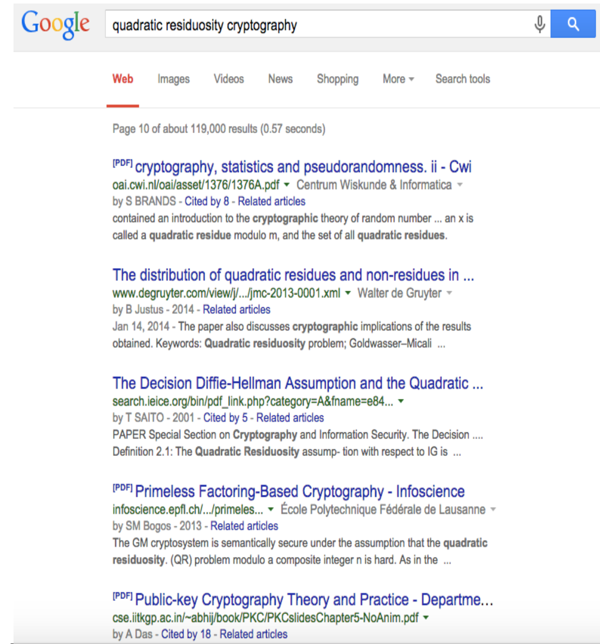
Quadratic Residuosity Assumption (QRA)

Let $QR_N(z) = \begin{cases} 0 \text{ if } z \text{ quadratic residue mod } N \\ 1 \text{ if } z \text{ is quadratic non-residue mod } N \end{cases}$

$\left(\frac{z}{N}\right) = 1$

QRA: ∀PPT algo A, ∀ n sufficiently large,
prob $(A(N,z) \neq QR_N(z)) >$ non-neg (n)
(over N, z where $\left(\frac{z}{N}\right) = 1$)

Quadratic Residuosoity is very Versatile



- Encryption:
  - Public Key Semantically secure
  - IBE [cocks]
  - Circular Security[BR]
  - Leakage Resilience [BR]
- Protocols:
  - Homomorphism : PIR [KO]
  - Interactive and Non-Interactive Zero Knowledge [GMR, BFM]
- First ZK protocol

Trapdoor Permutations

Is there a Collection of Trapdoor **Permutations** equivalent to factoring

Definition: Let I be a set of indices, and D_i a finite set. A collection of trapdoor permutations is a collection of one way permutations

$F = \{f_i : D_i \Rightarrow D_i\}_{i \in I}$

- Generation: $\exists$ PPT algorithm G that on input security parameter 1^n selects a random $f_i \in F$ with $|i|=n$ with short trapdoor information t_i
- Trapdoor-ness: $\exists$ PPT algorithm INV,

  s.t $INV(f_i(x), t_i) = x'$ such that $f_i(x)=f_i(x')$

Trapdoor Permutation Equivalent to Factoring [Blum-Williams]

Let N=pq, p,q primes s.t. p=q=3 mod 4

Define BW_N: $QR_N \rightarrow QR_N$ as $BW_N(x)=x^2 \bmod N$

Claim: When p=q=3 mod 4, then each quadratic residue mod N has a unique square root which itself is a quadratic residue mod N

Proof: (-1/p)=(-1/q)=-1 so -1 is a non-square.

So say root $x= c_1x_1+c_2x_2$ is a square, then x_1 is a square mod p and x_2 is a square mod q, which means that $-x_1$ and $-x_2$ are non-square mod p and q and thus all other roots of x^2 mod N are non-squares

Conclusion: BW_N is a permutation over the squares mod N

RSA:

Was the **first** example of Trapdoor permutation

Rivest-Shamir-Adelman
Turing Award

RSA Math

Let N=pq for p,q large prime and $\phi(N)=(p-1)(q-1)$
Let $e<\phi(N)$ such that $\gcd(e,\phi(N))=1$.
Ex: N=3*7=21, e=5, gcd(5,12)=1

Claim: Let $e<\phi(N)$ and d s.t. $de=1 \bmod \phi(N)$.
$\forall x$ in Z_n^*, $(x^e \bmod N)^d \bmod N = x^{ed \bmod \phi(n)} \bmod N = x \bmod N$

Define $RSA_{N,e}(x) = x^e \bmod N$ Ex: $2^5 \bmod 21 = 11$

Claim:. $RSA_{N,e} : Z_N^* \Rightarrow Z_N^*$ is a permutation

$RSA^{-1}_{n,e}(y) = y^d \bmod n : Z_N^* \Rightarrow Z_N^*$ where $e,d<\phi(N)$
$de=1 \bmod \phi(N)$

Proof: $(RSA_{n,e}(x))^d = x^{ed} \bmod N = x^{1 \bmod \phi(N)} \bmod n = x$

How hard is to generate N, e and d

- Choose p, q s.t. $|p|=|q|$ and set $N=pq$
- Choose e at random s.t. $\gcd(e, \phi(n))=1$
- Compute d s.t. $ed=1 \bmod \phi(n)$ using Euclidean-Gcd($e,\phi(n)$) to get d,c s.t.

$de+c\phi(n)=1$, and thus $de=1 \bmod \phi(n)$

How hard is to invert RSA given e and just N?

Claim: If can compute d, given N and e s.t. $ed=1 \bmod \phi(n)$, then can factor N

Proof: Homework

Does this mean that inverting RSA is as hard as Factoring?

Not necessarily. It may be possible to invert RSA without learning d and without factoring.

RSA and Factoring Integers

- Fact 1: Given N, e, p, and q, its easy to compute $\phi(N)$ and $d=e^{-1} \bmod \phi(N)$.
- Fact 2: Given only N,e, computing $\phi(N)$ is as hard as factoring N
- Fact 3: Given only N,e, computing d is as hard as factoring N
- Conclusions:
  - If can factor, can invert RSA
  - But, is Inverting (breaking) RSA as hard as factoring? MAJOR OPEN PROBLEM

RSA Assumption

$\forall$PPT algorithms *A*

Prob(*A*(N,e,x^e mod N) = x) < neg (n)

(over n-bit N=pq,

p,q primes of equal size

And e s.t. gcd(e,ϕ(N))=1 and $x \in Z_N^*$)

Strong RSA Assumption

$\forall$PPT algorithms *A*

Prob(*A*(N,y)=(e,x) s.t. y=x^e mod N) < neg (n)

(over n-bit N=pq,

p,q primes of equal size,

$y \in Z_N^*$)

If RSA is hard to invert in the worst case, it is hard to invert with non-neg probability

Claim: Fix N, $1<e<\phi(n)$.

If $\exists$ PPT B s.t. $prob_x(B(N,e,RSA_{N,e}(x))=x)>$ non-neg(n) then $\exists$ PPT algorithm A to invert $RSA_{N,e}(x)$ for all x.

Proof:

Given $y= x^e \bmod N$, choose random r in Z_N^* and map y to $z=y\, r^e \bmod N$. Now, run B(z). If successful, i.e $B(z)= xr \bmod N$, output $x= B(z)/r \bmod N$, else choose another r.

In expected $1/\varepsilon$ trials will be successful.

QED

RSA Collection of Trapdoor Functions

Define RSA = { $RSA_{N,e}$ } $_{N,e}$ where n=pq, for p,q primes s.t. |p|=|q|, $(e,\phi(N))=1$}

Theorem: Under RSA assumption,

RSA is a collection of trapdoor functions

Generation:

1. Choose at random n-bit p,q and test them for primality. If prime, set N=pq
2. Choose odd e, check that $gcd(e,\phi(N))=1$
3. Compute $d=e^{-1} \bmod \phi(N)$. d is the $trapdoor_{N,e}$

Evaluation: computing $RSA_{N,e}(x)$ takes $O(n^3)$ time

Hard to Invert: by RSA-Assumption

Trapdoor: Given N, e, and d, $x= (RSA_{N,e}(x))^d \bmod N$

Takes $O(n^3)$

Trapdoor Predicates

Trapdoor Predicates

A trapdoor predicate collection isa collection of Boolean functions $\{B_i: \{0,1\}^* \Rightarrow \{0,1\}\}_i$ s.t

- Easy to Generate Can generate (B_i,t_i) where ti is a trapdoor information
- Sample: For b $\in\{0,1\}$, there exists PPT algorithm A which outputs random s.t. $B_i(x)=b$
- Hard to Guess: For all PPT algorithms P, prob $(P(x)=B_i(x)) < ½$ +non-neg(n)
- Trapdoorness: there exist poly time algorithm Inv, s.t. $Inv(t_i,I, x)=B_i(x)$ for all x,i

- Where can we find trapdoor predicates?

Under QRA, $QR_N(z)$ is a trapdoor predicate for $N=pq$ for $p=q=3 \pmod 4$

Easy to Sample: $N=pq$ for $p=q=3 \pmod 4$

- Easy to sample in squares = $x^2 \bmod N$
- Easy to sample in non-squares with Jacobi symbol 1 = $-x^2 \bmod N$

- Where else can we find trapdoor predicates

Trapdoor Functions $\Rightarrow$ Trapdoor Predicates

Sample: Given b, choose x,r at random s.t. $\langle x,r\rangle=b$ and output $f'(x,r)=f(x),r$