April 19 - RSA

Logistics

- Project proposals due now!
- HW1 out now (due next week)
- Come to OHs!

Review from Last Time

Two ways to reason about cryptosystems in light of $P \stackrel{?}{=} NP$ open:

1) Make assumptions
2) Use idealized models for analysis.

Even Mansour Cipher

- Analysis uses random perm model
  - *all parties have oracle access to Π/Π^{-1}
  - *when use in practice, replace ideal Π w/ real $\hat{\Pi}$
    - ↳ heuristic security

$E: \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$

$E(k,m) := k \oplus \Pi(m \oplus k)$

Security: Used hybrid argument to show E is a PRP.

Game 0: Real attack game (adv talks to EM cipher)

Game 1: Rephrasing of real game

Game 2: Ideal game (in which adv talks to real random permutation)

April 19 - RSA

Review II

Time/Space Trade-Offs & Hellman Tables

Application: "Inverting functions with advice."

$[N] = \{1, \ldots, N\}$.

Say you are given $f : [N] \to [N]$, $y \in [N]$ and S bits of "advice" about f. Your task: find $x \in [N]$ s.t. $y = f(x)$.

Thm (Hellman): With $S \in (N^{2/3})$ advice bits, can invert f in time $O(N^{2/3})$.

[Note: inverting a random f given oracle access to it takes $\Omega(N)$ time!]

Consequence: After precomputation, inverting DES takes only $\approx 2^{40}$ time.

Idea: Build a table that is essentially a compressed version of f.

Collision-Finding on $f : [N] \to [N]$

- Meet in the middle: Space $O(\sqrt{N})$, Time $O(\sqrt{N})$ ↘!
- Rho method: Space $O(1)$, Time $O(\sqrt{N})$
- Parallel w p procs: Space $O(1)$ (per proc), Time $O(\sqrt{N}/p)$ ↗!

April 19 - RSA and Number-Theoretic Crypto

- RSA was originally interesting b/c was first construction of digital signature, PKE scheme
  → We know now that you can build these from discrete log assumption, others
- RSA is also interesting because it has more structure than d.log..
  ↳ more functionality! (accumulators, later in lecture)

Going out of style for two reasons
1) Public keys & signatures are large → elliptic-curve crypto
2) Quantum computers break RSA (also break ECC)

A Survey of Hard Problems

FACTORING: Sample $p, q \xleftarrow{R} \{\lambda\text{-bit primes}\}$
$N \leftarrow pq$
Given N, produce (p, q).

- We will discuss factoring algs later on in qtr.
- Best known alg runs in time

$$\approx e^{O(\lambda^{1/3}(\log \lambda)^{2/3})}$$ "General number field sieve" (Pollard 1988)
← Not polynomial in λ!

RSA Factoring assumption: FACTOR is hard! (Different from random integer!)

RSA-e Problem (e is odd prime)
Sample $p, q \xleftarrow{R} \{\lambda\text{-bit primes}\}$ s.t. $\gcd(p-1, e) = \gcd(q-1, e) = 1$. (Why?)
$N \leftarrow pq$.
$x \xleftarrow{R} \mathbb{Z}_N$
$y \leftarrow x^e \in \mathbb{Z}_N$ { fancy notation for "mod N"
Given (N, y) produce x.

Review from CS255:

$\mathbb{Z}_N = \{0, \ldots, N-1\}$
$\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \mid \gcd(x, N) = 1\}$ = "set of invertible elements in $\mathbb{Z}_N$"

For an RSA modulus $N = pq$.
$|\mathbb{Z}_N^*| = \phi(N) = (p-1)(q-1)$. } See textbook/CS255 slides
For prime p, $\exists$ generator $g \in \mathbb{Z}_p^*$ s.t.
$\mathbb{Z}_p^* = \{g^0, g^1, g^2, g^3, \ldots, g^{q-1}\}$ and $\forall x \in \mathbb{Z}_p^*$ $x^q = 1 \in \mathbb{Z}_p^*$

RSA

The "RSA-e assumption" asserts that computing e-th roots modulo N is hard.

→ often take $e = 3$, $e = 65537 = 2^{16}+1$ (Why?)

Crazier Assumptions

Strong-RSA Problem

Sample $p, q \xleftarrow{R} \{\lambda\text{-bit primes}\}$

$N \leftarrow p \cdot q$

$a \xleftarrow{R} \mathbb{Z}_N$

Given (N, a) produce (x, e) s.t.

$a = x^e \in \mathbb{Z}_N$ and $e \neq \pm 1$.

Strong RSA assumption is that computing eth root, for any e of adversary's choosing is hard.

Hardness Relation

FACTORING $\geq$ RSA $\geq$ Strong RSA

Note that:
- No one knows if an alg for solving RSA problem yields an alg for factoring N
- Ditto for Strong RSA → RSA
- RSA-e problem has a unique answer
- Strong RSA problem has exponentially many answers
  - * seems .... stronger!
  - * Still, best known alg is to factor N
  - * Not convinced that this is best alg.

RSA Random Self-Reduction

For a given modulus N, we would like that $a^{1/e} \bmod N$ is hard to compute for "almost all" $a \in \mathbb{Z}_N$.

"Hard on average"

We know that for some $a \in \mathbb{Z}_N$, computing $a^{1/e} \bmod N$ is easy!
Is it possible that computing $a^{1/e} \bmod N$ is easy for $\frac{1}{100}$ fraction of a's? No!
We can show that either:

* Finding $a^{1/e} \bmod N$ is hard for almost all N, or
* Finding $a^{1/e} \bmod N$ is easy everywhere

} No middle ground.

Claim Say there exists an eff alg A_N s.t.

$$\Pr_{a \leftarrow \mathbb{Z}_N}\left[A_N(a) = a^{1/e} \in \mathbb{Z}_N \right] = \epsilon$$

then there exists an eff alg B_N s.t. for <u>all</u> $x \in \mathbb{Z}_N$

$$\Pr_{\substack{\text{randomness} \\ \text{of } B_N}}\left[B_N(x) = x^{1/e} \in \mathbb{Z}_N \right] = \epsilon$$

Pf $B_N(x)$ {

$r \xleftarrow{R} \mathbb{Z}_N$

$y \leftarrow A_N(x \cdot r^e)$

$z \leftarrow y \cdot r^{-1} \in \mathbb{Z}_N$.

if $z^e \neq x$: output "fail"

else output z.

}

$$\Pr[\overline{\text{fail}}] = \Pr_a\left[A_N(a) = a^{1/e} \in \mathbb{Z}_N \right] = \epsilon$$

→ Can iterate B many times to amplify success prob

Note: The random self-red is for a fixed N.
↳ Taking e-th roots mod N is hard for most $a \in \mathbb{Z}_N$.
is it's hard for any $a \in \mathbb{Z}_N$.

An open question is whether it's possible to construct R.S.R. between factoring problems:
↳ If factoring is easy for an ϵ fraction of 2λ-bit RSA moduli, is it hard for almost all? No idea

Crypto from Factoring

Trapdoor OWF (Review from CS255)

$(pk, sk) \leftarrow Gen(1^\lambda)$

$y \leftarrow F(pk, x) \qquad x \in \mathcal{X},\ y \in \mathcal{Y}$

$x \leftarrow F^{-1}(sk, y)$

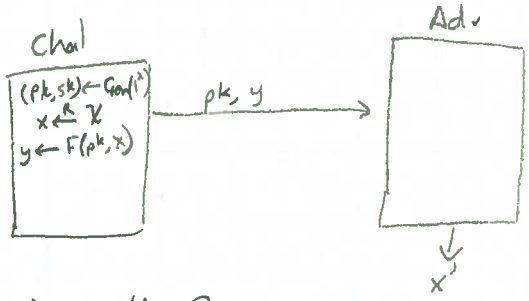
Correctness

For all (pk, sk) from Gen, for all $x \in \mathcal{X}$, $F^{-1}(sk, F(pk, x)) = x$.

Security

For all eff adv A,

$TDFAdv[A, F] := Pr[x = x'] \in negl(\lambda)$.



From TDF, can build PKE. (CS255). How?

Rabin (1979)

- RSA is cool, but requires a new assumption
- Build TDF from factoring (not RSA)

Idea: $(N, p) \leftarrow Gen(1^\lambda)$ // return RSA modulus

$y \leftarrow F(N, x)$

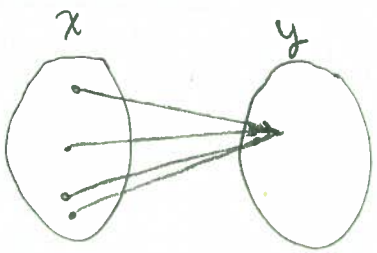
return $y = x^2 \in \mathbb{Z}_N$

$x \leftarrow F^{-1}(p, y)$

compute square root of $x \in \mathbb{Z}_N$

At a high level: Rabin is RSA but with $e = 2$.

↳ Unlike RSA, here F is <u>not</u> a permutation



Chinese Remainder Thm (Reminder from CS 255)

Given p, q s.t. $\gcd(p,q) = 1$, given x_p and x_q s.t.

$$x_p = x \pmod{p}$$
$$x_q = x \pmod{q}$$

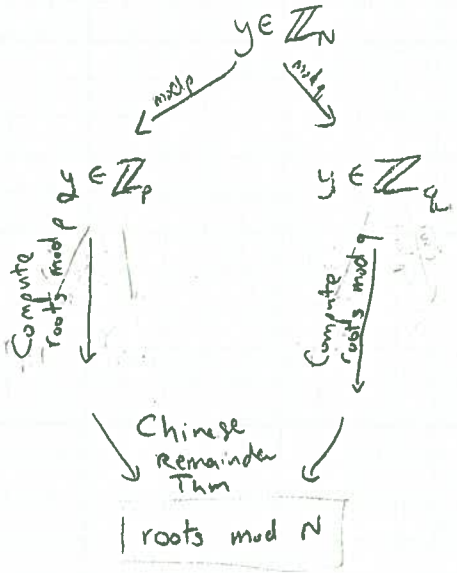
there is an eff. alg that outputs $x \pmod{pq}$.

PF See textbook.

Rabin

Issues:

1) How do we take square roots mod N?



Note: This process requires knowing factors of N!

Also, we need to take square roots mod p, q.

Claim: If $p \equiv q \equiv 3 \pmod 4$ then $x = y^{\frac{p+1}{4}}$ is a square root of y in $\mathbb{Z}_p^*$.

pf

$$x^2 = \left(y^{\frac{p+1}{4}}\right)^2 = y^{\frac{p+1}{2}} = y \cdot y^{\frac{p-1}{2}} = y\left(r^{p-1}\right)^{\frac{p-1}{2}} = y \cdot \underbrace{r^{p-1}}_{\text{must be } 1} = y \in \mathbb{Z}_p^*$$

So, if $p \equiv q \equiv 3 \bmod 4$, taking roots is easy!

Also easy if p or $q \equiv 1 \bmod 4$ .... Tonelli-Shanks alg.

Analysis of Rabin

Claim: $y \in \mathbb{Z}_N^*$ has four square roots in $\mathbb{Z}_N$, if it has any.

First see that $y \in \mathbb{Z}_p^*$ has 2 square roots if it has any:

PS Let x be s.t. $x^2 = y \in \mathbb{Z}_p^*$.

Then $(p-x)^2 = p^2 - 2px + x^2 \in \mathbb{Z}_p^*$

$= x^2 \in \mathbb{Z}_p^*$

$= y$

[Just as $4 = 2^2 = (-2)^2$, $y = x^2 = (-x)^2 \in \mathbb{Z}_p^*$]

Now, looking at roots mod $N = pq$

If $y = x_p^2 \in \mathbb{Z}_p^*$ and $y = x_q^2 \in \mathbb{Z}_q^*$

then using the Chinese Remainder Thm, can compute x s.t. $x^2 = y \in \mathbb{Z}_N^*$. But now we have 4 roots in $\mathbb{Z}_N^*$:

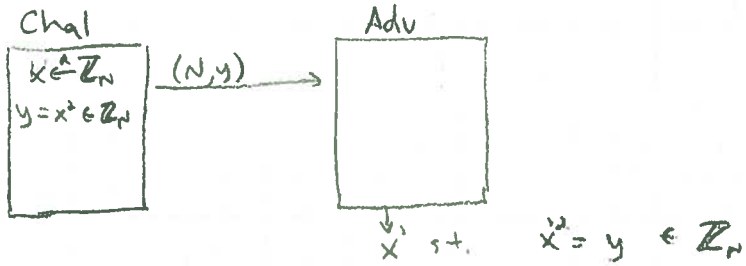
(x_p, x_q), $(x_p, -x_q)$, $(-x_p, x_q)$, $(-x_p, -x_q)$

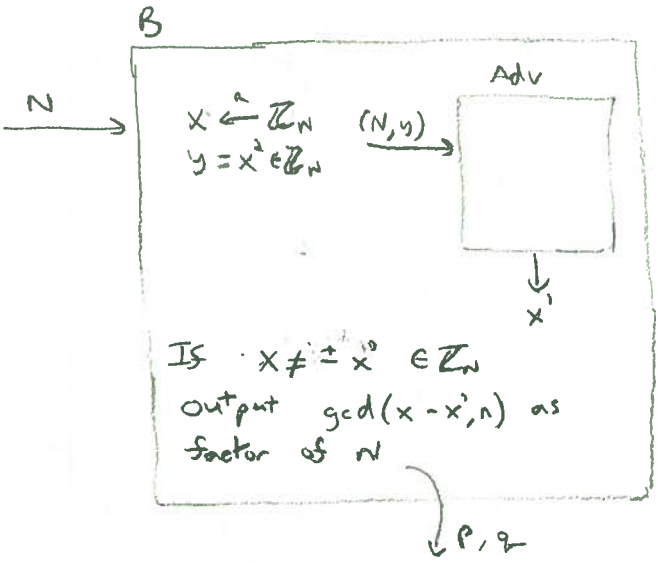
These are the inverses of $x^2 \in \mathbb{Z}_N^*$ under the Rabin function.

Rabin

Security: * This is the brilliant idea.
* The fact that each $y = x^2 \in \mathbb{Z}_n$ has many inverses is key to the proof
↳ contrast w/ RSA.

PS Idea Use adv A inverting Rabin's function w.p ϵ to produce alg B that factors N. with probability $\frac{\epsilon}{2}$.





Rabin

* We have x and x' s.t.

$$x^2 = x'^2 \pmod N$$
$$x^2 - x'^2 = 0 \pmod N$$
$$(x+x')(x-x') = 0 \pmod N$$

IF $x \neq \pm x'$, then

$$\underbrace{(x+x')}_{\text{multiple of } p}\underbrace{(x-x')}_{\text{multiple of } q} = kN \in \mathbb{Z}$$

This idea is the core of almost all factoring algs!

To complete the proof, must show that

$$Pr[x \neq \pm x'] = \frac{1}{2}.$$

Idea: There are four equally likely possibilities

$x = x' \pmod p$	$x = \pm x' \pmod q$	} Not useful
$x = -x'$	$x = x'$	} Useful
$x = x'$	$x = -x'$	
$x = -x'$	$x = -x'$	} Not useful

$\Rightarrow$ w.p. $\frac{1}{2}$, we end up in a useful case and can factor N.

Another View of RSA Assumptions

Let N be an RSA modulus.

(Rabin) For $a \xleftarrow{R} \mathbb{Z}_N$, find root of $f(x) = x^2 - a \in \mathbb{Z}_N$.

(RSA^e) For $a \xleftarrow{R} \mathbb{Z}_N$, find root of $f(x) = x^e - a \in \mathbb{Z}_N$.

What about

(Crazy RSA) For $a \xleftarrow{R} \mathbb{Z}_N$, find root of $f(x) = x^7 + 4x^3 + a \in \mathbb{Z}_N$?

Or

For $a \xleftarrow{R} \mathbb{Z}_N$, find root of $f(x,y) = x^7 + 3y^7 + a \in \mathbb{Z}_N$?

$\Rightarrow$ In general, if you know factors of N, all of these problems are easy.

- Solve mod p, mod q
- Get sol'n mod N using CRT.

$\Rightarrow$ If you <u>don't</u> know factors, then only apparent way is to solve over rationals and reduce mod N.

e.g. $f(x) = x^3 - 8$ mod N is easy to solve w/o p, q.

Generating Random Primes in Practice

Alg 1(1^λ): do { choose random $x \xleftarrow{R} [2^{\lambda-1}, 2^\lambda)$
while (x is not prime)
return x.

Running Time: By Prime # Thm, $\pi(2^\lambda) = \#\text{primes} \leq 2^\lambda \approx \frac{2^\lambda}{\ln(2^\lambda)}$

$$\pi(2^\lambda) - \pi(2^{\lambda-1}) \approx \frac{2^\lambda}{\lambda} - \frac{2^{\lambda-1}}{\lambda-1} \approx \frac{2^{\lambda-1}}{\lambda^2}$$

Probability that a random x is prime is then

$$\frac{\#\text{primes}}{2^{\lambda-1}} \approx \frac{2^{\lambda-1}}{\lambda 2^{\lambda-1}} \approx \frac{1}{\lambda}$$

So alg will terminate after $\approx \lambda$ iterations.

For a primality test can use Miller-Rabin test
↳ randomized