Great Theoretical Ideas In Computer Science
Venkat Guruswami
CS 15-251 Fall 2014
Lecture 18 Oct 28, 2014 Carnegie Mellon University

CRYPTOGRAPHY

Cryptography: A land of counterintuitive possibilities

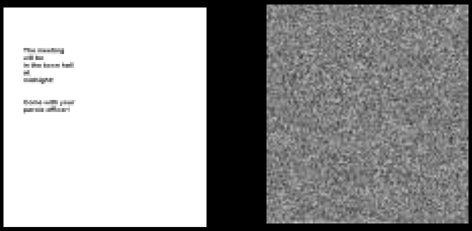
- Alice and Bob can agree on a secret key over a public channel
- Alice can convince Bob she knows something – say proof of twin prime conjecture – with Bob learning nothing about the proof
- Anyone can publicly send an encrypted message to Bob that only he can decrypt, without any pre-agreed upon secret

Cryptography: A land of counterintuitive possibilities

- One can delegate computation of any function on encrypted data without revealing anything about the inputs
- Millionaires' Problem: Alice and Bob can find out who has more money without revealing anything else about their worth
- One can learn a piece of data from a database without the database learning anything about your desired query
- ….

Private/Symmetric Key Encryption

One Time Pads

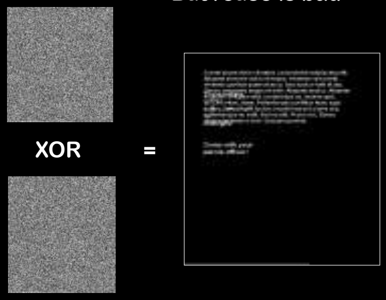


Add (XOR) a secret key, shared between sender &receiver, to the message.

One Time Pads

Gives perfect security!
For random shared key,
leaks no information about message

But reuse is bad


XOR

=

Agreeing on a secret

One time pads rely on having a shared secret!

Need a separate secret for each pair of communicating parties.

Does this require private communication to agree on the secret?

Can Alice and Bob agree on a secret over a public conversation?

Diffie-Hellman Key Exchange

Alice: Picks prime p, and a generator g in Z_p*
Picks random number a $\in$ {1,2,…,p-1}
Sends over p, g, g^a (mod p) to Bob

Bob: Picks random b $\in$ {1,2,…,p-1} and sends over g^b (mod p) to Alice

Now both can compute the shared "secret"
g^{ab} (mod p)

It's good there are hard problems!

Alice: Picks prime p, and a value g in Z_p*
Picks random a in {1,2,…,p-1}
Sends over p, g, g^a (mod p)

Bob: Picks random b in {1,2,…,p-1}, and sends over g^b (mod p)

Secret: g^{ab} (mod p)

Given g^a , a is uniquely determined.
So why is this secure?

Discrete Log intractability assumption:
Given input a large prime p, g in Z_p*, and x=g^a,
it is hard to compute a (= $\log_g$ x)

Crypto needs hard problems to keep bad guys at bay (privacy)
But good guys should be able to achieve desired functionality
This delicate balance is the challenge and beauty of crypto

Hard algebraic problems

Hardness to keep bad guys at bay (privacy)
Easiness for good guys to operate (functionality)

Algebra (groups, number theory) is a great source of problems meeting these demands.

What about Eve?

Alice: Picks prime p, and a value g in Z_p*
Picks random a in {1,2,…,p-1}
Sends over p, g, g^a (mod p)

Bob: Picks random b in {1,2,…,p-1}, and sends over g^b (mod p)

Secret: g^{ab} (mod p)

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

Diffie-Hellman assumption computing g^{ab} (mod p) from p, g, g^a, g^b is hard

To say Eve learns *nothing* about the shared secret (eg. its first bit) need g^{ab} (mod p) to be look like a random element of Z_p*
(This is *Decisional Diffie-Hellman* (DDH) assumption;
not valid for Z_p* but there are other candidate cyclic groups)

Why these assumptions?

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

Finding discrete logarithms seems hard, but *proving* the hardness seems even harder!

Proving intractability of Discrete-Log is harder than the P vs. NP problem

Complexity-theoretic cryptography relies on assumptions on the presumed intractability of some (classes) of problems.

- Information-theoretic crypto: no hardness assumptions (eg. one time pad)

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

Can we get rid of this assumption?

Can someone who I have never spoken to send me a message over a public channel that is only intelligible to me

---

Public Key Encryption

Public Key Encryption
[Diffie-Hellman]

Goal: Enable Alice to send encrypted message to Bob *without their sharing any secret*

Anyone should be able to send Bob a message in encrypted form.

Only Bob should be able to decrypt.

Anyone can send Bob a message in encrypted form. *Only* Bob should be able to decrypt.

HOW ???

Bob has to be "special" somehow…

Bob holds a special "secret key" that only he knows and that enables him to decrypt

- (Hopefully) decryption intractable without knowledge of this secret.
- Physical analogy: key to a locked box

Bob holds a "secret key" (known only to him) that enables him to decrypt

- Physical analogy: key to a locked box

Encryption (Physical analogy):

- Place message in a locked box with a "lock" that only Bob's key can open.

How to get hold of such lock(s)?

Bob "gives them" to everyone!!

Bob has a ***"public key",*** known to everyone, which can be used for encryption.

Public Key Encryption

Pair of fns. (Enc,Dec) for encryption & decryption

Bob generates a (PK,SK) pair.
- Publishes PK.
- Holds on to SK as a secret

Encryption of message m: Enc(m, PK)
- Anyone can encrypt (as PK is public)

Decryption of ciphertext c: Dec(c, SK)
- Bob knows SK so can decrypt.

Of course, must have Dec(Enc(m,PK),SK) = m

Take 1

Alice, who has never spoken to Bob, wants to send him message m in encrypted form Enc(m)

Recovering m from Enc(m) should be a hard problem

How about Enc(m) = g^m mod p
(where g,p are public knowledge)

Discrete log hardness ⇒privacy from eavesdropper

But how will Bob figure out m ??
- He has to solve the same discrete log problem!
- Seems tricky to give him an egde

The RSA Cryptosystem

Modular Arithmetic
Interlude #3

Modular arithmetic

Defn: For integers a,b, and positive integer n,
$a \equiv b \pmod{n}$ means
(a-b) is divisible by n, or equivalently
a mod n = b mod n (x mod n is remainder of x when divided by n, and belongs to {0,1,…,n-1})

Fundamental lemmas mod n:

Suppose $x \equiv y \pmod{n}$ and $a \equiv b \pmod{n}$. Then
1) $x + a \equiv y + b \pmod{n}$
2) $x * a \equiv y * b \pmod{n}$
3) $x - a \equiv y – b \pmod{n}$

So instead of doing +,,- and taking remainders, we can first take remainders and then do arithmetic.*

~~Fundamental lemma of powers?~~

If $x \equiv y \pmod{n}$
Then $a^x \equiv a^y \pmod{n}$?

NO!

$2 \equiv 5 \pmod{3}$, but it is not the case that:
$2^2 \equiv 2^5 \pmod{3}$

(Correct) rule for powers.

If $a \in Z_n^*$ and $x \equiv y \pmod{\phi(n)}$
then $a^x \equiv a^y \pmod{n}$

Equivalently, for $a \in Z_n^*$, $a^x \equiv a^{x \bmod \phi(n)} \pmod{n}$

Euler's theorem: for $a \in Z_n^*$, $a^{\phi(n)} \equiv 1 \pmod{n}$

If $x = q\,\phi(n)+r$,
Then $a^x = a^{q\,\phi(n)}\, a^r \equiv a^r \pmod{n}$

Example…

$5^{121242653}$ (mod 11)

121242653 (mod 10) = 3

5^3 (mod 11) = 125 mod 11 = 4

Why did we take mod 10?

$343281^{327847324}$ mod 39

Step 1: reduce the base mod 39

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

NB: you should check that gcd(343281,39)=1 to use lemma of powers

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

---

RSA prepwork I: computing in Z_n^*

Computing in Z_n^*

- Multiplication: easy, just multiply mod n
- Exponentiation: To compute a^m, do "repeated squaring" ≈ $\log_2$ m multiplies mod n
- Inverses: To compute a^{-1}
- use extended Euclid algorithm to compute r,s such that r a + s n = 1.
- Then a^{-1} = r mod n.

How do you compute…

5^8 using few multiplications?

First idea:

5 5^2 5^3 5^4 5^5 5^6 5^7 5^8

= 5*5 5^2*5

How do you compute…

5^8

Better idea:

5 5^2 5^4 5^8

= 5*5 5^2*5^2 5^4*5^4

Used only 3 mults instead of 7 !!!

Repeated squaring calculates

a^{2^k}

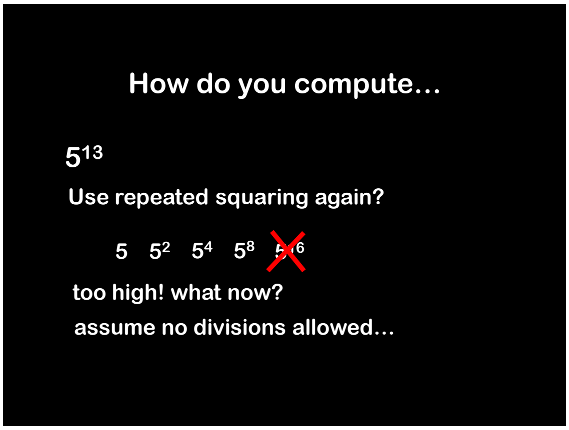
in k multiply operations

compare with

$(2^k - 1)$ multiply

operations

used by the naïve method



How do you compute…

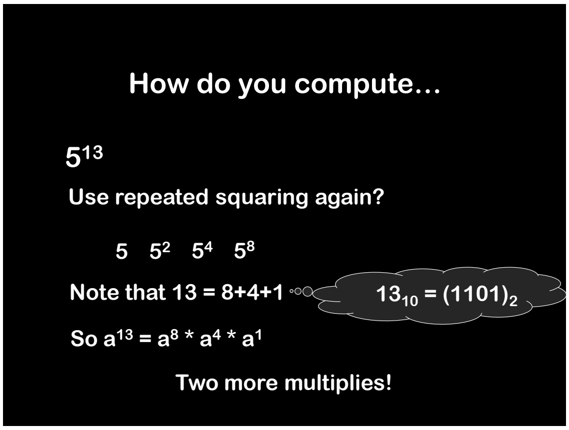
5^{13}

Use repeated squaring again?

5 5^2 5^4 5^8 $\cancel{5^{16}}$

too high! what now?

assume no divisions allowed…



How do you compute…

5^{13}

Use repeated squaring again?

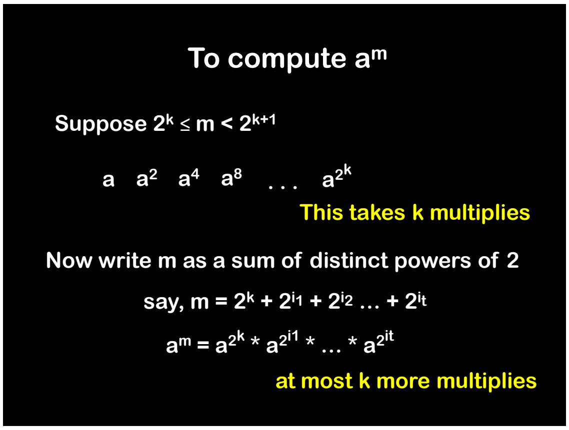
5 5^2 5^4 5^8

Note that 13 = 8+4+1

$13_{10} = (1101)_2$

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

Two more multiplies!



To compute a^m

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

a a^2 a^4 a^8 . . . a^{2^k}

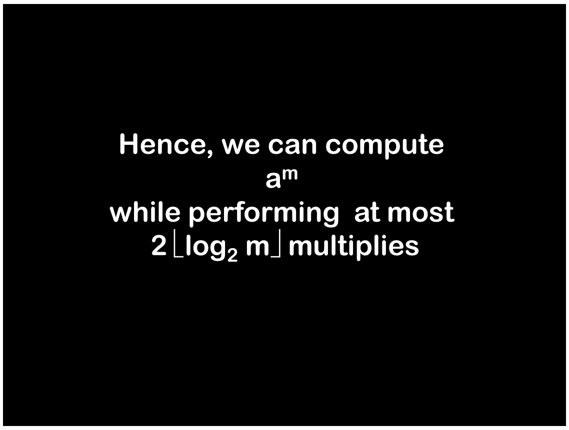
This takes k multiplies

Now write m as a sum of distinct powers of 2

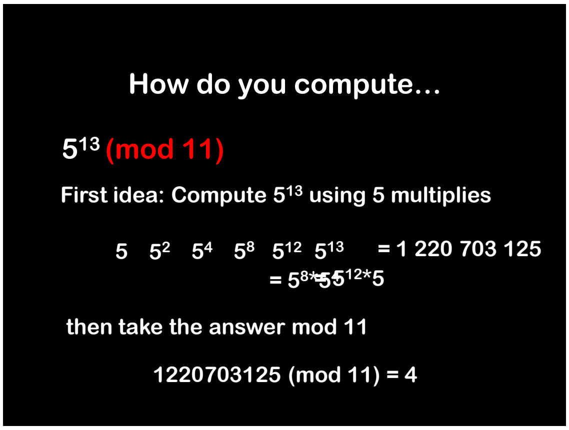
say, $m = 2^k + 2^{i_1} + 2^{i_2} \ldots + 2^{i_t}$

$a^m = a^{2^k} * a^{2^{i1}} * \ldots * a^{2^{it}}$

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} (mod 11)

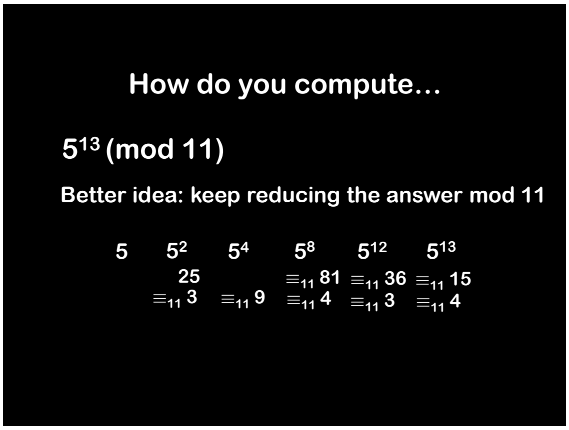
First idea: Compute 5^{13} using 5 multiplies

5 5^2 5^4 5^8 5^{12} 5^{13} = 1 220 703 125

= 5^8*5^4 $5^{12}*5$

then take the answer mod 11

1220703125 (mod 11) = 4



How do you compute…

5^{13} (mod 11)

Better idea: keep reducing the answer mod 11

5	5^2	5^4	5^8	5^{12}	5^{13}
	25		$\equiv_{11}$ 81	$\equiv_{11}$ 36	$\equiv_{11}$ 15
	$\equiv_{11}$ 3	$\equiv_{11}$ 9	$\equiv_{11}$ 4	$\equiv_{11}$ 3	$\equiv_{11}$ 4

Hence, we can compute
a^m (mod 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

$Z_{15}^* = \{1 \leq x \leq 15 \mid \gcd(x,15) = 1\}$
$= \{1,2,4,7,8,11,13,14\}$

$\phi(15) = 8$

*	1	2	4	7	8	11	13	14
1	1	2	4	7	8	11	13	14
2	2	4	8	14	1	7	11	13
4	4	8	1	13	2	14	7	11
7	7	14	13	4	11	2	1	8
8	8	1	2	11	4	13	14	7
11	11	7	14	2	13	1	8	4
13	13	11	7	1	14	8	4	2
14	14	13	11	8	7	4	2	1

RSA prepwork

Theorem: If p,q are *distinct* primes then
$\Phi(pq) = (p-1)(q-1)$

Proof: We need to count how many numbers in
{1,2,3,…,pq-1} are relatively prime to pq.

Let us count those that are not, and subtract from (pq-1).

These are

(i) the multiples of p:	p, 2p, 3p, …, (q-1)p
(ii) the multiples of q:	q, 2q, 3q, …, (p-1)q

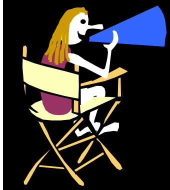
Total = q-1 + p-1=p+q-2

So $\Phi(pq) = pq-1 - (p+q-2) = pq-p-q+1 = (p-1)(q-1)$

RSA Cryptosystem

**Rivest, Shamir, Adleman:
2002 A.M. Turing Award**

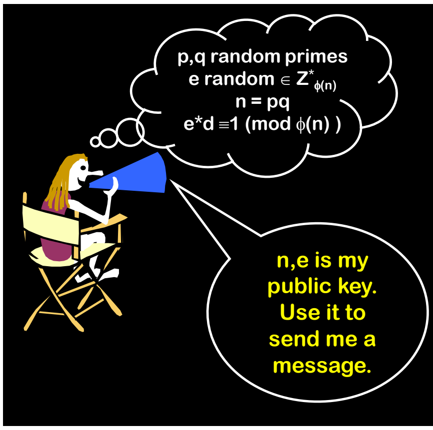
The RSA Cryptosystem



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

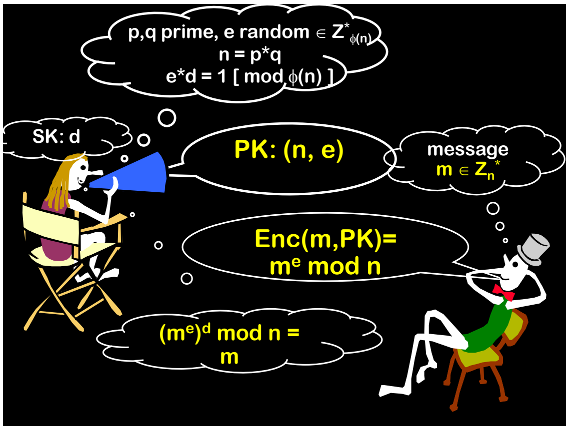
$\phi(n) = \phi(p)\,\phi(q) = (p-1)*(q-1)$
Pick random $e \in Z^*_{\phi(n)}$
"Publish": e

Compute d = inverse of e in $Z^*_{\phi(n)}$
Hence, $ed \equiv 1 \pmod{\phi(n)}$
"Private/secret Key": d



p,q random primes
e random $\in Z^*_{\phi(n)}$
n = pq
$e*d \equiv 1 \pmod{\phi(n)}$

n,e is my
public key.
Use it to
send me a
message.



RSA: Simple example

How hard is breaking RSA?

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

If we can compute Φ(n) from n, can we crack RSA?

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

We don't know this…

What does (breach of) security mean?

Certainly complete recovery of m by bad guys

But also learning partial information about m

- eg. value of m (say salary info) up to +/- $1000

How to define security to capture the requirement that **no information** about m is leaked?

Information-theoretic perfect secrecy

For the one time pad solution, the eavesdroppers have no clue about m, regardless of computing power

- *The distribution of ciphertexts does not depend on m*
- Say adversary knows either m_0 or m_1 was sent, and sees the ciphertext.
  - Still can't tell which of m_0 or m_1 was sent better than 50-50 guessing
  - Thus seeing the ciphertext has *no* bearing on adversaries abilities to learn m

What about computational security in Public Key Encryption?

Great Definitions & Solution Concepts: Semantic Security, Probabilistic Encryption

Goldwasser, Micali: 2012 Turing Award

Both Ph.D. advisees of now CMU Professor Manuel Blum

Semantic Security

> Given ciphertext and message length, adversary cannot determine any partial information about the message with success probability non-negligibly larger than when he only knows the message length (but not the ciphertext)

Equivalent to following:

- Let m_0 and m_1 be any two messages of equal length (known to all).
- Adversary is presented Enc(m_b, PK) for random b
- The adversary shouldn't be able to find b with probability non-negligibly better than 50-50

Probabilistic Encryption

Semantic security: Adversary shouldn't be able to tell apart Enc(m_0,PK) from Enc(m_1,PK)

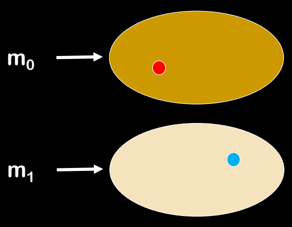
But anyone (including the adversary) can compute Enc(m, PK) from m ….

How can Enc(m, PK) hide m in above strong sense?

Have many possible encryptions for each m
Enc(m, PK) should be a *randomized encryption* of m

Probabilistic Encryption

Enc(m, PK) = random ciphertext from many possible encryptions



Knowing SK allows recovery of m from Enc(m,PK)

Adversary can't tell apart random red point from random blue point

Goldwasser-Micali Public Key Encryption

- **Probabilistic encryption scheme**
- **Semantically secure under certain "quadratic residuosity" intractability assumption (which is related to hardness of factoring)**

Key Generation

1. Pick large primes p,q with p,q ≡ 3 (mod 4)
2. Compute n = pq

Public Key: n
Secret Key: p,q

Remark: Integers n of above from are called Blum integers (after CMU professor Manuel Blum)

For a Blum integer n,
(n-1) is a quadratic non-residue
which means $x^2 \equiv (n-1) \pmod n$ has no solutions

Encryption by Alice

Scheme encrypts bits (for longer messages, break into bits and apply encryption to each bit separately)

Enc(b, PK=n):

1. Pick a random $y \in Z_n^*$
2. Output $(n-1)^b y^2$

Fact: Enc(b,n) is a quadratic residue mod n if and only if b=0

Decryption by Bob

Ciphertext c=Enc(b,n) is a quadratic residue mod n (i.e., $\exists$ x s.t $x^2 \equiv c$ (mod n)) if and only if b=0

How can Bob (who has the secret key) determine if c is a quadratic residue mod n

Bob's advantage: He knows the factors p,q of n

Exercise 1: c is a quadratic residue mod n if and only if c is a quadratic residue modulo *both* p, q

Exercise 2: c is quadratic residue mod prime p if and only if $c^{(p-1)/2} \equiv 1$ (mod p)

Eavesdropping by Eve

What does the adversary see?

Enc(b,n) = $(n-1)^b y^2$ (mod n) for a random $y \in Z_n^*$

For encryption of bit 0,

- a random quadratic residue mod n

For encryption of bit 1,

- a random quadratic non-residue* mod n

* actually random quadratic non-residue c such that n–c is a quadratic residue (mod n)

Semantically secure?

Given large n = pq with unknown factorization, it is believed that distinguishing random quadratic residues from random quadratic non-residues is hard

This assumption implies semantic security of the GM scheme

Remark (nice exercise): Finding square roots of quadratic residues modulo n=pq enables finding the prime factors p,q of n

The Elgamal Encryption Scheme

- **Another probabilistic public key encryption scheme**
- **Based on hardness of Discrete Log (Diffie-Hellman assumption)**

The Elgamal Encryption Scheme

- **Public key: prime p, generator g of Z_p^*, and $h = g^x \bmod p$**
- **Private key: x ($\in \{1,2,\ldots,p-1\}$)**

Encryption: To encrypt $m \in Z_p^*$:

- **Pick $y \in \{1,2,\ldots,p-1\}$ at random**
- **Output (g^y mod p , m h^y mod p)**

Public key: prime p, generator g & $h = g^x \bmod p$
Private key: x

Encryption: To encrypt $m \in Z_p^*$:

- Pick $y \in \{1,2,\ldots,p-1\}$ at random
- Output $(g^y \bmod p,\ m\ h^y \bmod p)$

Decryption: To decrypt (c_1,c_2) with private key x:

- Compute $s = c_1^x \bmod p$
  (this is the "shared secret" for this message)
- Output $m = c_2 s^{-1} \bmod p$

Theorems about breaking Elgamal

If discrete log is easy, then easy to decrypt

Assuming that g^{xy} is hard to compute given g,g^x,g^y (CDH assumption), encryption is hard to invert.

Assuming one can't tell apart g^{xy} from a random element even when given g,g^x,g^y (DDH assumption), stronger "semantic security".

Operating on Ciphertexts

For RSA, given ciphertexts encrypting m_1 and m_2, one can compute ciphertext encrypting the product $m_1\ m_2$ (i.e., there is no need to decrypt, can directly multiply in the encrypted world)

- $(m_1m_2)^e \equiv m_1^e\ m_2^e \pmod n$

Same holds for Elgamal scheme also

- $(g^y,\ m_1h^y) * (g^{y'},\ m_2h^{y'}) = (g^{y+y'},\ m_1m_2\ h^{y+y'})$

For Goldwasser-Micali, one can compute encryption of $b \oplus b'$ given ciphertexts for b and b'

$$(n-1)^b\ y^2\ (n-1)^{b'}\ z^2 \equiv (n-1)^{b \oplus b'}\ (yz)^2 \pmod n$$

Partially malleable encryption

These encryption schemes allow us to perform either addition or multiplication directly on ciphertexts.

Rivest, Adleman, Dertouzos 1978 wondered:
Is there an encryption scheme that would allow one to ***both*** add and multiply within the encrypted world?

They foresaw that such a completely malleable encryption scheme allowing arbitrary computations on encrypted data would have amazing applications (eg. today think of delegating computation to the cloud without revealing your inputs)

However, finding such a plausible scheme, which these days we call "fully homomorphic encryption" (FHE) remained open for over 30 years

Craig Gentry in 2009 gave the first candidate FHE scheme

[Picture from his 2014 MacArthur Fellowship announcement]

Very high level & sketchy idea behind approach:

- Encrypt by noisy encoding of message as per some error-correcting code
- Decrypt by removing noise (which requires a secret nice representation of the code)
- Add and multiply operations increase the noise by a small amount
- When noise gets too large, "refresh" ciphertext

Curious? : see this beautiful recent survey:
https://eprint.iacr.org/2014/610

Non-malleable encryption

Sometimes, we actually *don't* want ciphertexts to be malleable

- Eg. if you are submitting bidding D dollars in encrypted form, you don't want someone to encrypt (D+1) dollars based on your bid

Candidates of such non-malleable encryption schemes are also known (starting with Dolev, Dwork, Naor 1991)

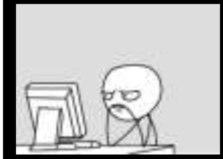
Summary

Cryptography is a field with a variety of questions and challenges, rich underlying theory, and profound applications.

It hinges on structured hard computational problems

Algebra and number theory form a fertile ground of such problems

---



Study Guide

One time Pad

Diffie-Hellman Key Exchange

Public Key Cryptography

Modular arithmetic:
Fundamental lemma of powers

RSA encryption

Probabilistic encryption

Goldwasser-Micali and Elgamal encyption schemes

Supplementary Material: Primality Testing

How do we generate large primes?

These encryption schemes require large prime numbers

- eg. n = p q for RSA for large primes p,q

Primes are reasonably dense (1/n of n-bit numbers of are prime).

$\Rightarrow$ If we can efficiently test if a number is prime, then we can generate primes fast (by selecting a few random numbers and checking those for primality).

Answer: The Miller-Rabin primality test [1976,80]

(Gary Miller is one of our professors.)

Primality testing

Given n, output if it is prime.

Naïve method: Try all numbers 2,3,…,$\sqrt{n}$

If any of them divide n, output NO,

else output YES.

Problem: Huge runtime (think of n as a 500 digit number…)

Goal: To do this efficiently, with runtime scaling well with # digits of n.

Primality testing

Given n, output if it is prime.

By Fermat's little theorem, if n is prime,

$a^{n-1} \equiv_n 1$ for all a in [1, n-1].

So, here's a possible test:

- pick random a $\in$ {1,2,…,n-1}
- Check if $a^{n-1} \equiv_n 1$, if so output prime, else output composite.

1. Can repeat with say 50 random choices of a.
2. If n is prime, algo. definitely outputs prime.

Primality testing

1. pick random $a \in \{1,2,\ldots,n-1\}$; if $\gcd(a,n) \neq 1$, output composite
2. If $a^{n-1} \equiv_n 1$, output prime, else output composite.

Key to analysis: Number of "witnesses" $a \in Z_n^*$ satisfying $a^{n-1} \neq 1 \bmod n$ *when n is not prime.*

Lemma: *If just one witness exists, then in fact at least ½ the a's must be witnesses.*

Proof: Let $w \in Z_n^*$ be such that $w^{n-1} \neq 1 \bmod n$.

Define $B = \{ b \mid b^{n-1} \equiv_n 1 \}$ (these are the "non-witnesses")

Key observation: If $b \in B$, then $w b \notin B$.

Injection from B to B^c. So $|B^c| \geq |Z_n^*|/2$.

Carmichael numbers

Unfortunately, there are composite numbers n, called Carmichael numbers, such that

$$a^{n-1} \equiv_n 1 \text{ for all } a \in Z_n^*$$

In fact, there are infinitely many of them, smallest one being 561 = 3 * 11 * 17

On these numbers, our Fermat's Little Theorem based test fails ☹

The Miller-Rabin test gives a correct version that works for all n.

Idea behind Miller-Rabin test

- Another way to prove n is composite is to exhibit a *"fake square root"* modulo n
  - x such that $x^2 \equiv 1$ (mod n), but x mod n $\neq \pm 1$
- Why does such an x prove that n is not prime?

Miller-Rabin test

Say we write n-1 as $d * 2^s$ where d is odd.

Consider the following sequence of numbers mod n:

$$a^d , a^{2d}, a^{4d}, \ldots , a^{d*2^{(s-1)}}, \quad a^{d*2^s} = a^{n-1} \equiv_n 1$$

Each element is the square (mod n) of the previous one.

If n is prime, then at some point the sequence hits 1 and stays there from then on.

What is the number right before the first 1 ?

If n is prime this MUST BE n-1. (WHY?)

$$a^d , a^{2d}, a^{4d} . . . a^{d*2^{(s-1)}}, a^{d*2^s} = a^{n-1} \equiv_n 1$$

If n is prime, then at some point the sequence hits 1.

The number before the first 1 must be n-1

Miller-Rabin Test

pick a random a and generate the above sequence.

If the sequence does not hit 1, then n is composite.

If there's an element before the first 1 and it's not n-1, then n is composite.

Otherwise n is "probably prime".

Theorem (we won't prove this):

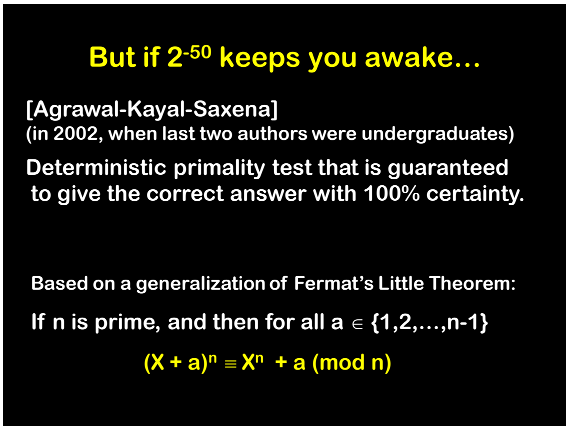
If n is composite, at least ½ the a's "catch" its compositeness via the above test.

Miller-Rabin Analysis

If n is composite, then with a random a, the Miller-Rabin algorithm says "composite" with probability at least ½ (in fact at least ¾)

So if we run the test 50 times and it never says "composite" then n is prime with "probability" $1-2^{-50}$

In other words it's more likely that you'll win the lottery three days in a row than that this is giving a wrong answer.



But if 2^{-50} keeps you awake…

[Agrawal-Kayal-Saxena]
(in 2002, when last two authors were undergraduates)

Deterministic primality test that is guaranteed to give the correct answer with 100% certainty.

Based on a generalization of Fermat's Little Theorem:

If n is prime, and then for all $a \in \{1,2,\ldots,n-1\}$

$$(X + a)^n \equiv X^n + a \pmod{n}$$