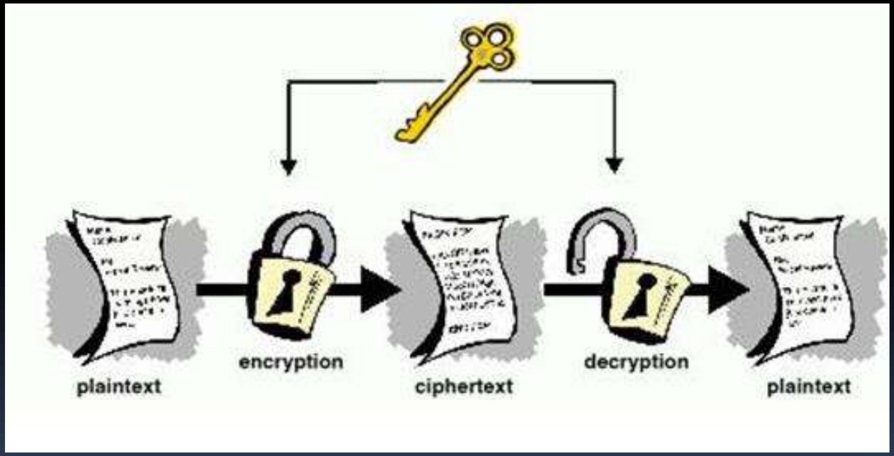
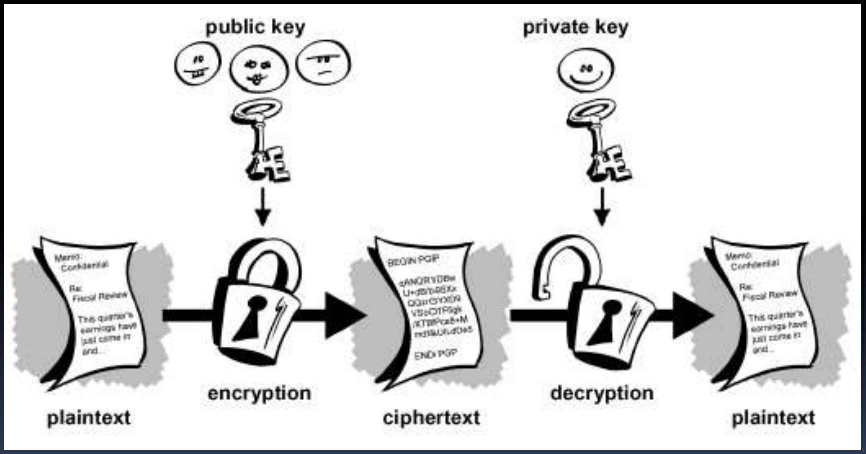
15-251: Great Theoretical Ideas in Computer Science

Fall 2016 Lecture 23

November 15, 2016

Cryptography





Cryptography: A land of counterintuitive possibilities

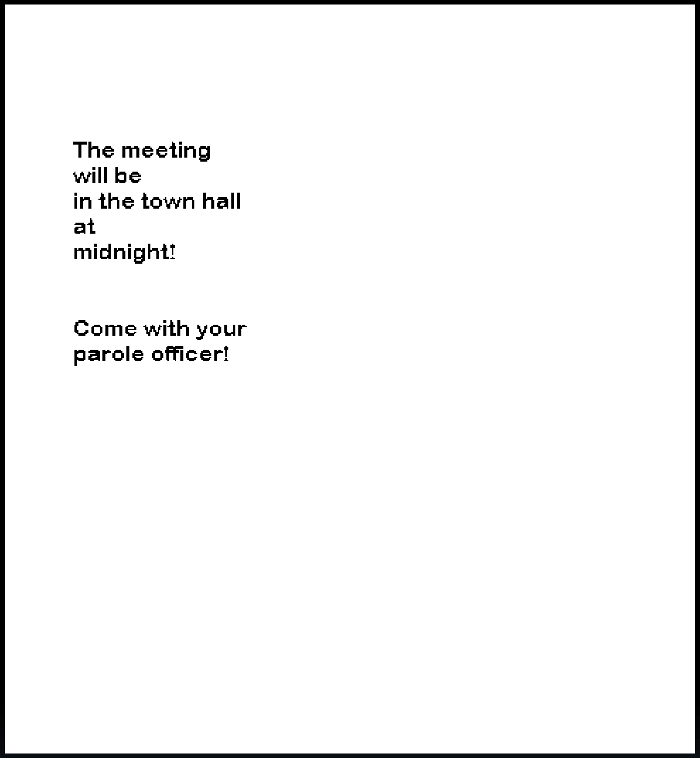
- Alice and Bob can agree on a secret key over a public channel
- Alice can convince Bob she knows something – say proof of twin prime conjecture – with Bob learning nothing about the proof
- Anyone can publicly send an encrypted message to Bob that only he can decrypt, without any pre-agreed upon secret

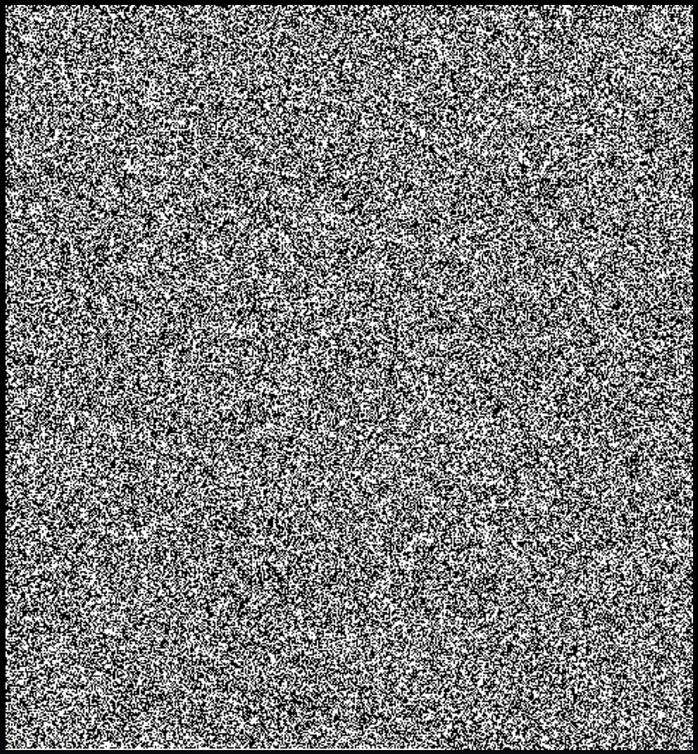
Cryptography: A land of counterintuitive possibilities

- 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

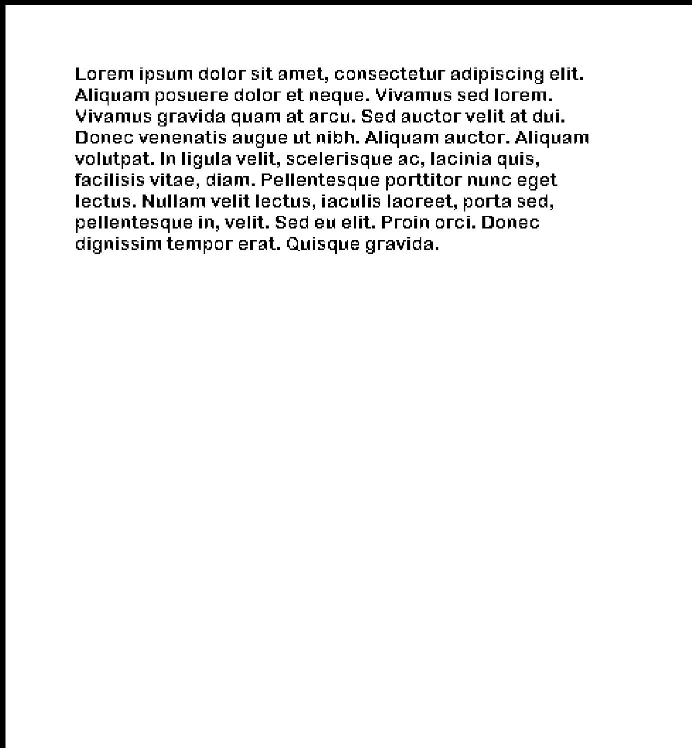
One Time Pads



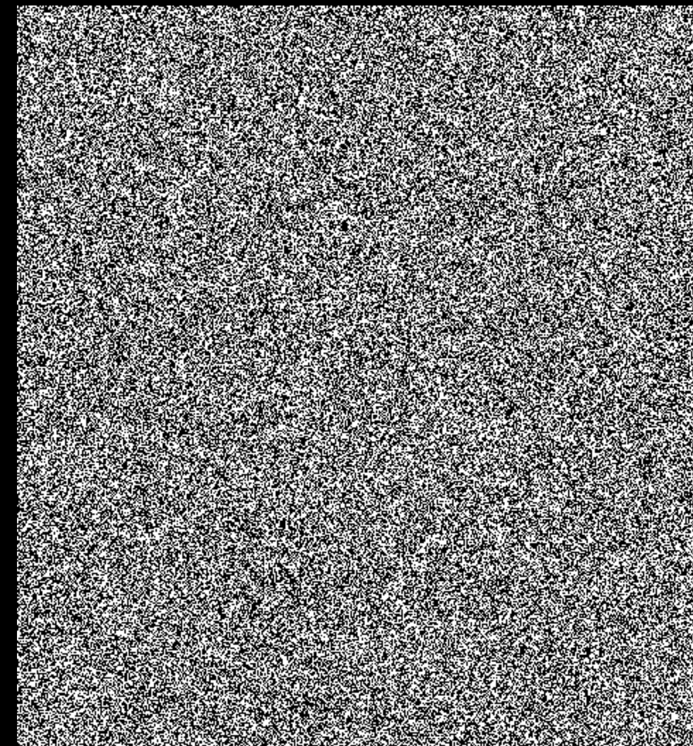


Add (XOR) a secret key, shared between sender & receiver, to the message.

One Time Pads



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam posuere dolor et neque. Vivamus sed lorem. Vivamus gravida quam at arcu. Sed auctor velit at dui. Donec venenatis augue ut nibh. Aliquam auctor. Aliquam volutpat. In ligula velit, scelerisque ac, lacinia quis, facilisis vitae, diam. Pellentesque porttitor nunc eget lectus. Nullam velit lectus, iaculis laoreet, porta sed, pellentesque in, velit. Sed eu elit. Proin orci. Donec dignissim tempor erat. Quisque gravida.



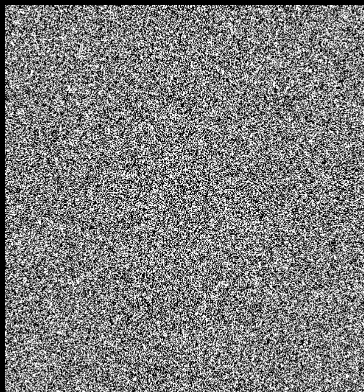
$$Enc_k(m) = m \oplus k$$

$k =$ shared secret key

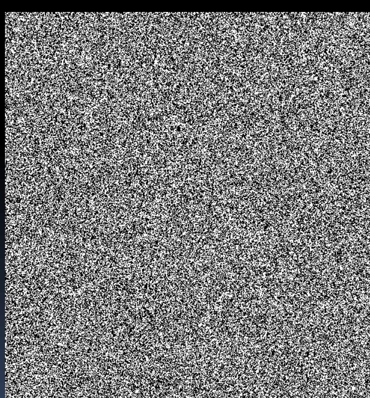
$$Dec_k(c) = c \oplus k$$

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

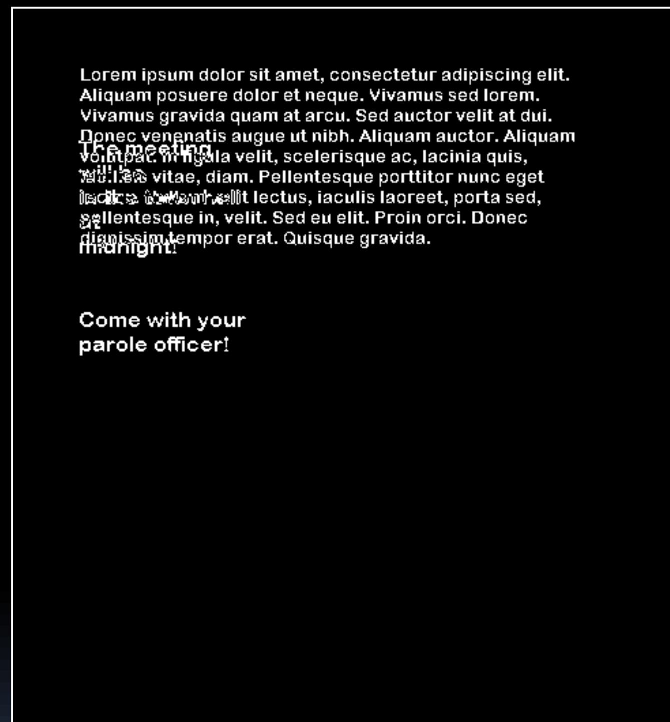
But reuse is bad



XOR



=



$$Dec_k(c_1 \oplus c_2) = m_1 \oplus m_2$$

Encryption of one known message allows one to recover key k

One time pad needs a shared secret key as large as the total number of bits to be communicated!

This is ***necessary*** for perfect secrecy! ☹

But if we relax security from *"information-theoretic"* (against *arbitrary* eavesdroppers) to *"computational"* (against, say, *polynomial time* eavesdroppers), then we can do much better (under suitable intractability assumptions)!

"complexity-theoretic cryptography"

A Great Idea: Pseudorandomness

Computational lens on randomness:
Something is "pseudorandom" if a polytime adversary can't distinguish it from a pure random string

Pioneered by Blum-Micali'84; Yao'82

1995

2012

2000

[Year of A. M. Turing Award]

Pseudorandomness: A Peek

Basic primitive: A *pseudorandom generator* (PRG)

Deterministic map $G : \{0,1\}^n \to \{0,1\}^{n+1}$

"Creates" one extra bit of randomness

For random $r \in \{0,1\}^n$, $G(r)$ *looks random*
(even though it clearly is not: it's supported on half of $\{0,1\}^{n+1}$)

Such a map can be iterated (in a suitable manner)
to increase the stretch

Alice and Bob can share a secret key $r \in \{0,1\}^n$, stretch it to n^2 bits, and encrypt n messages $\mathrm{m_i} \in \{0,1\}^n$, using the i'th block of $G(r)$ as "one-time pad" for m_i

A hard to compute bit is pseudorandom

If $G(r)$ is random looking, must be hard to predict last bit from from first n bits.

How about $G(r) \coloneqq r \circ h(r)$,
where $h: \{0,1\}^n \to \{0,1\}$ is a *hard function*
(for random input r, hard to predict $h(r)$ better than 50-50)

<u>Example</u>: $h : Z_p^* \to \{0,1\}$ defined as $h(y) = 1$ iff $\log_g y > \frac{p-1}{2}$ where g is a generator of Z_p^* & $\log_g(\cdot)$ is the discrete logarithm to base g (inverse of the map $x \mapsto g^x$)

We believe this function is hard for large p and most generators of Z_p^*

Computing the PRG

$G(r) \coloneqq r \circ h(r)$ (for random input r, hard to predict $h(r)$ better than 50-50)

Output of G is pseudorandom, but…

G itself is hard to compute!

Another great idea:
Use "one-way easiness" of some functions
(eg. multiplication is easy, but factoring seems hard)

$G(x) = g^x \circ half(x)$

$half(x) = 1$ iff $x > \frac{p-1}{2}$

Easy to compute + output looks random (no clue about half(x) based on g^x)

Agreeing on a secret

- Private key cryptography relief on parties having a shared secret (even in PRG based scheme)

- 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 via a *completely public conversation*?

NO WAY, right?

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!

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

Alice: Picks prime p, and a generator 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)

Discrete Log intractability assumption:

Given input a large prime p, g in Z_p^*, and $y=g^a$,
it is hard to compute a ($= \log_g y$)

Crypto needs hard problems to keep bad guys at bay (security)
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 (security/privacy)
Easiness for good guys to operate (functionality)

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

What about eavesdropping Eve?

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

Alice: Picks prime p, and a generator 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)

To say Eve learns *nothing* about the shared secret (eg. its first bit), need g^{ab} (mod p) to be pseudorandom (look like a random element of Z_p^*)

(This is the *Decisional Diffie-Hellman* (DDH) assumption;
not quite true in 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)

One step further

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* in a manner that is *only intelligible to me*

Public Key Encryption

2015
A.M. Turing Award

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 everybody, which can be used for encryption.

Public Key Encryption

Pair of functions (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) = $g^m \bmod p$
(where g,p are public knowledge)

Discrete log hardness $\Rightarrow$ 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 edge

Key exchange to Public Key Encryption?

Recall the Diffie-Hellman key-exchange protocol
(I've swapped Alice & Bob's roles)

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

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

Secret: g^{ab} (mod p)

Idea: Instead of sending p, g, g^b (mod p) just to Alice, Bob publishes this as his public key PK!

- Keeps b as his secret key SK

To encrypt m, Alice uses g^{ab} as a (multiplicative) one-time pad

~~Enc(m,PK)=m g^{ab}~~

Bob needs g^a to learn the mask g^{ab}

Have Alice include it in the encryption!

Enc(m,PK)= (g^a , m g^{ab})

The ElGamal Public Key Encryption Scheme [1985]

- Public key: prime p, generator g of Z_p^* & $h = g^b \bmod p$
- Private key: b $(\in \{1,2,\ldots,p-1\})$
- Encryption: To encrypt $m \in Z_p^*$:
  - Pick $a \in \{1,2,\ldots,p-1\}$ at random
  - Output $(g^a \bmod p, m h^a \bmod p)$

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

- Compute $s = c_1^b \bmod p$

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

Comments on ElGamal Scheme

Astonishing that it took 8+ years from the Diffie-Hellman key exchange protocol to the encryption scheme

- In fact, this was not the first proposal for a PKE
- That honor belongs to the RSA scheme (1976)

Security of encryption scheme based on same assumption as key exchange:
given (p, g, g^a, g^b) it is hard to compute g^{ab} in Z_p^*

The encryption scheme is *randomized*

- *This is not a bug, it is a necessary feature*
- Without randomness, there is no secure cryptography

The RSA Cryptosystem

Modular Arithmetic Interlude (oh no, not again!)

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?

$343281^{327847324} \bmod 39$

Step 1: reduce the base mod 39

Step 2: reduce the exponent mod $\Phi(39) = 24$

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: computing in Z_n^*

Computing in Z_n^*

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

Modular exponentiation

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

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

$\boxed{\phi(15) = 8}$

*	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

<u>Theorem</u>: If p,q are *distinct* primes then

$\Phi(pq) = (p-1)(q-1)$

Poll

<u>Proof</u>: We need to count how many numbers in $\{1,2,3,\ldots,pq-1\}$ are relatively prime to pq.

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

These are

(i) the multiples of p: $p, 2p, 3p, \ldots, (q-1)p$

(ii) the multiples of q: $q, 2q, 3q, \ldots, (p-1)q$

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

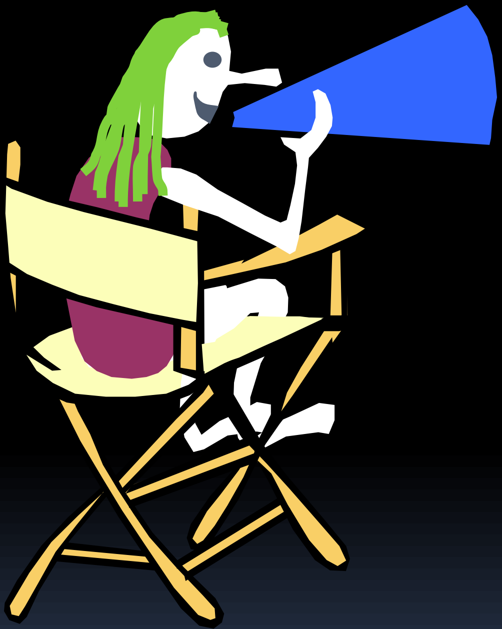
So $\Phi(pq) = pq-1 - (p+q-2) = pq-p-q+1 = (p-1)(q-1)$

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

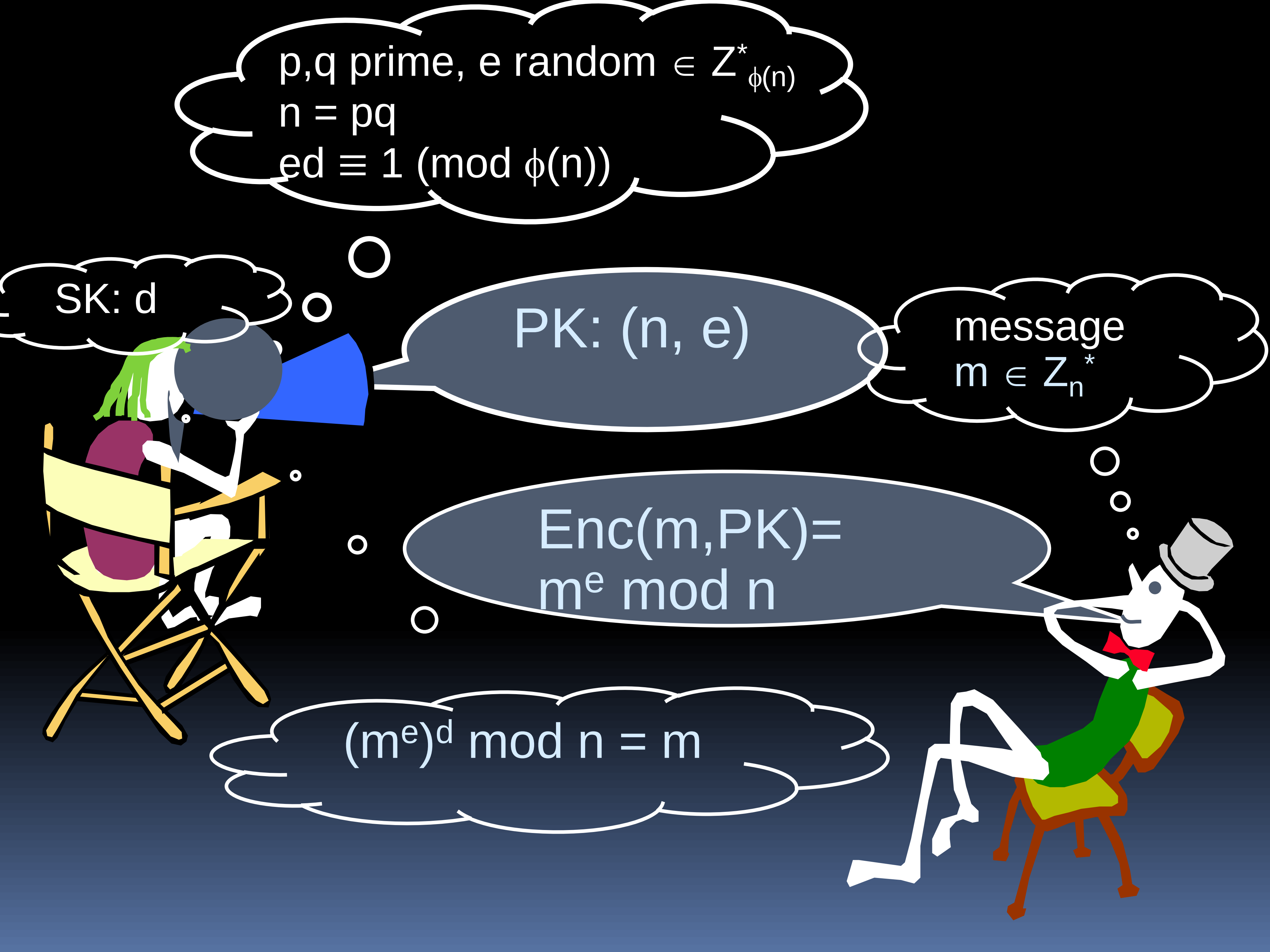
p,q random primes

e random $\in Z^*_{\phi(n)}$

$n = pq$

$ed \equiv 1 \pmod{\phi(n)}$

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


p,q prime, e random $\in Z^*_{\phi(n)}$
$n = pq$
$ed \equiv 1 \pmod{\phi(n)}$
SK: d
PK: (n, e)
message
$m \in Z_n^*$
$Enc(m,PK)=$
$m^e \bmod n$
$(m^e)^d \bmod n = m$

RSA: Simple example

$n = 3 \times 11 = 33$

$e = 3$

Public key: $(33,3)$

What is the RSA ciphertext c encrypting $m = 13$?

$\phi(n) = 20$

$d = 3^{-1}\, mod\, 20 =$

Private key $d = 7$

What is the decryption of $c = 19$?

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 doesn't 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 adversary's ability to learn

For computational security (based on pseudorandom pads), no polytime adversary can predict if m_0 or m_1 was sent better than ½ + negl(n)

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.
Shafi Goldwasser
also a CMU undergrad.

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

Need randomness as in ElGamal scheme

Security of RSA & ElGamal Schemes

Is RSA encryption scheme semantically secure
(knowing that either m_0 or m_1 was encrypted,
is it hard to guess which one was encrypted)?

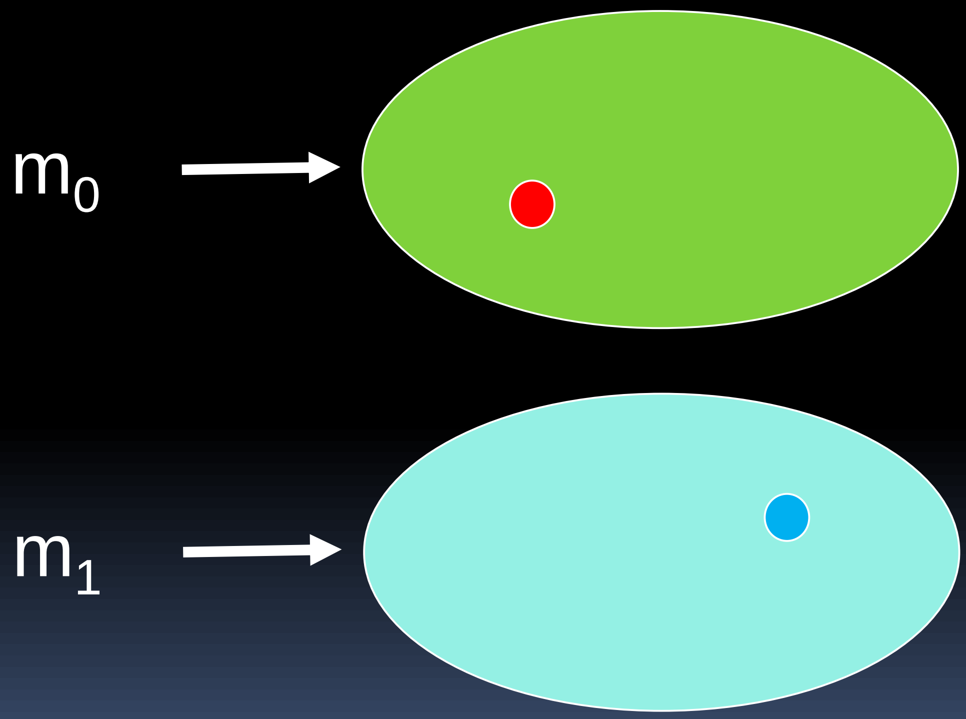
- No! The encryption is deterministic.
- But there are probabilistic variants which are secure if $x^e \, mod \, n$ is very hard to invert

Is ElGamal scheme sematically secure?

- Yes, under Decisional Diffie-Hellman assumption in concerned group
  - $\left(g, g^a, g^b, g^{ab}\right)$ is indistinguishable from (g, g^a, g^b, h) where h is an independent uniform group element

Probabilistic Encryption

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



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

Adversary shouldn't
be able to tell apart
random red point
from
random blue point

Goldwasser-Micali Public Key Encryption Scheme

- 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 $\equiv$ 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)

Fact: For a Blum integer n,
(n-1) is a quadratic non-residue (non-square) modulo n
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 $(n-1)^b y^2 \in Z_n^*$

<u>Note</u>: Enc(b,n) is a quadratic residue (square) modulo 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 $x^2 \equiv c \pmod{n}$)   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 $c^{(p-1)/2} \equiv 1 \pmod{p}$

Eavesdropping by Eve

What does the adversary see?

$\mathrm{Enc}(b,n) = (n-1)^b\, y^2 \pmod{n}$ 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

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)

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

Same holds for Elgamal scheme:

- $(g^a, m_1 h^a) * (g^{a'}, m_2 h^{a'}) = (g^{a+a'}, m_1 m_2 h^{a+a'})$

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

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

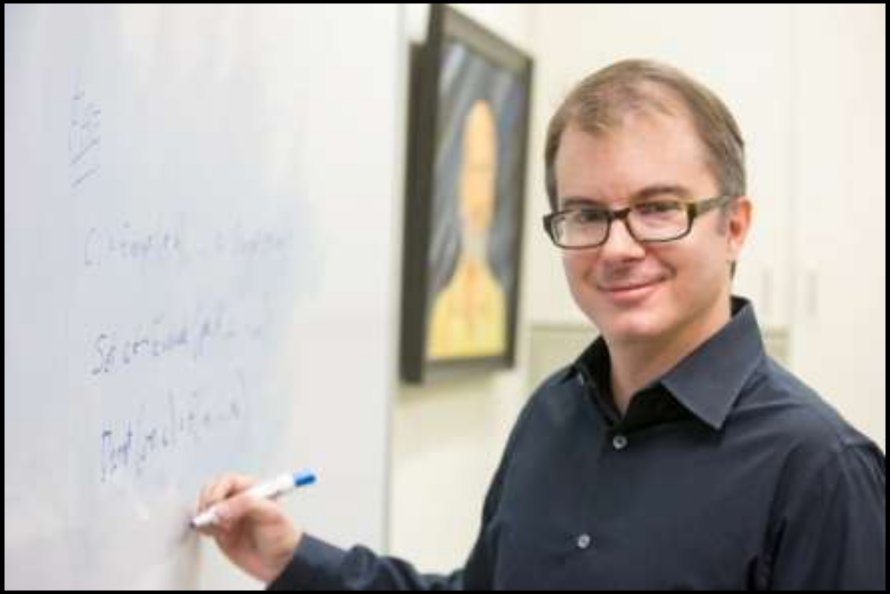
Partially malleable encryption

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 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 survey "Computing on the edge of chaos": 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 (or 1.1 D dollars) based on your bid

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

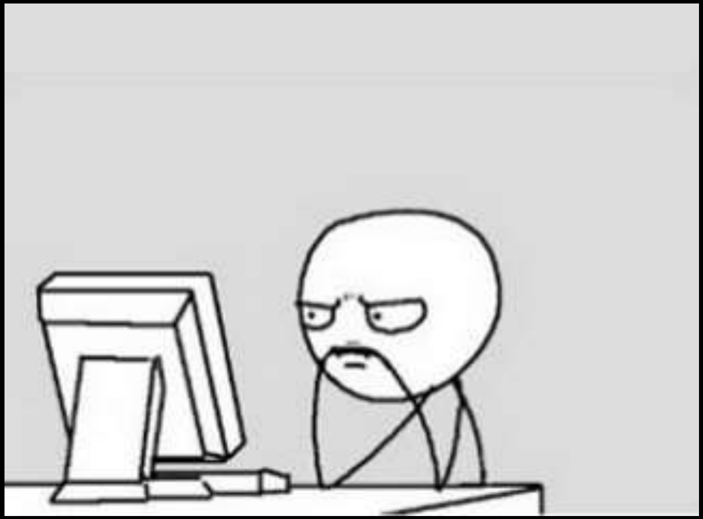
One of our newly hired faculty members
(Vipul Goyal, arriving Jan 2017)
is an expert on non-malleable cryptography

Summary

Cryptography is a field with a
host of challenges (that seem impossible at first blush),
conceptually deep definitions,
rich underlying theory,
and profound applications.

It hinges on structured hard computational problems

Algebra and number theory are a
fertile source of such problems



Study Guide

One time Pad

Pseudorandomness (informal)

Diffie-Hellman Key Exchange

Public Key Cryptography

ElGamal public key encryption

RSA encryption scheme

Probabilistic encryption

Goldwasser-Micali publick key encryption scheme