CSC 350 Cryptology and Security

Lecture 1

Review Syllabus
Bookmarks with delicious. You each take a week, contribute bookmarks that week.

Three parts (part 1 and 2 of the book)

- Classical encryption – Pre computers
- Modern encryption
- Public Key encryption

Plus security projects from you. Topics posted within a month. (part 3 and 4 of the book)

Three **goals** with security:

- Confidentiality - keep secrets from those who should not know
- Integrity – know who sent it for real, and that the message is the real message
- Availability – the information can be found by those who need to know

Alice and Bob are our main characters. They want to communicate in secret, but Eve is trying to listen in on the conversations.

Bank Example: What to keep confidential?

- Integrity?
- Availability?

Attacks on security

- Confidentiality:
  - Snooping watching packet contents
  - Traffic analysis watching packet patterns
- Integrity:
  - Modification change transmission
  - Masquerading pretending to be someone you're not
  - Replaying sending a message again for benefit : make bank give money
  - Repudiation not Eve, but Alice or Bob denying that a message happened
- Availability:
  - Denial of Service flood a server with tons of requests for information

Bank example again? How could these happen to a bank?

Passive vs active attacks. Passive tries to find out information, active harms things

- Which are passive and which are active?

Security **Services**

- Tangible things people want with security, more than the 3 above

Data Confidentiality
Data Integrity   anti-change, anti-replay
Authentication   peer entity, data origin
Nonrepudiation   proof of origin, delivery
Access Control

Security **Mechanisms**

How the services are implemented:

Encipherment   cryptography and steganography
Data integrity   checksums
Digital signature   electronically sign and verify signature
Authentication exchange   prove identities to each other
Traffic padding   bogus traffic
Routing control   continuously changing routers to prevent eavesdropping
Notarization   third party control
Access control   proof of access. Passwords, PINs.

Lecture 2 Steganography

Perplex City cards

5
22
25
135
175
226

Others

- Shifting letters up and down, from old typewriter you expect this…
- First letter of each sentence.
- Make a template

LSB

- A picture is worth 1000 words they say.
- It depends on the image size… ☺

- Messages we want to send are written in strings.
- Each string is made up of characters
- Each character is a number
- Each number can be written in binary
  - To make one large binary number string

- An image is a collection of pixels
- Each pixel is a tuple of red, green and blue.
- Where can we hide our binary numbers?
  - Change the shades of the color very slightly.
  - Almost imperceptible to human eye
  - Computer eye too? (file size important)

- This method works well with lossless compression schemes. (GIF, PNG)
- How does it work? Encode:
  - Look at the color of each pixel.
  - If you want to represent a 1, make it an odd number for that color,
  - Otherwise make it an even number.

- Decode:
  - Look at each pixel, if odd, write out a 1, if even, write out a zero.
  - Turn these numbers back into ascii characters and generate the string

- Example
  - Encode "HELLO"
  - 0110100001100101011011000110110001101111

Then change the bits in the image

Demo

Show file .png

Show encoded .png

Use program to decode image

gifshuffle

or how many colors are in the image

how does a gif work.

There's a color table, and each pixel points to an entry in that table.

The ordering of that color table can hold log2(n!) bits, where n is the number of colors in the table.

Lecture 3 – Number Theory, Modular Arithmetic

Questions about Lab 1?

Topics

- Integers
- Binary Operations
- GCD / Extended Euclid
- Mod is binary operator
- Inverses
- Sets with Zp

The basics behind cryptography: you have information. Letters, bits, etc. You need a way to transform that data to be unrecognizable, but for the process to be easily reversible if you know the right key values. It must be hard to attack this algorithm from the outside.

These transformations have a lot of mathematics going on underneath them. Today, integers and mod.

Integers, set called Z. Zahlen (German for numbers)
Binary operations on integers:

- Addition
- subtraction
- multiplication
- Take two input, give you one output, which is in the set Z

Division does not fit this definition. It gives us two outputs, quotient and remainder

- We restrict this to make remainder positive #.
- Java does not do this. Take away divisor once more. q-1
    - Add the divisor once more to the result.
- -255 / 11 = -23 * 11 + -2 or -24 * 11 + 9   9 is the positive remainder

divisibility notation |, |/

- if a = q * x, no remainder, then x|a x divides a (opposite in book, typo, download errata from book website)
- if it's divisible, how big can can you go?
- leads us to greatest common divisor GCD, great for small, how about large?
- how to find GCD?
    - Bad alg: try all numbers starting smallest of a, b, descending.
    - Euclidean algorithm.
    - 1) gcd(a, 0) = a (why? Greatest divisor of a is a, any number divides 0 into 0, without mod.)
    - if a and b are divisible by x, then so are sum and difference.
        - Let a = x*y and b = x*z
        - Sum is x*(y+z), diff is x*(y-z)

Gcd(24, 16) = gcd(16, 8) = gcd(8, 0) = 8
So, whatever divides 24 and 16, must divide 24-16 = 8.
we keep this up, until this subtraction is less than our original smaller number. This is the remainder of division.

2) gcd(a, b) = gcd(b, r) where r is the remainder of dividing a by b

what kind of definition is this? Recursive.
Let's write it up in python

```
def gcd(a, b):
	if b == 0:
		return a
	else:
		return gcd(b, a % b)
```

book has iterative loop with while.

Also, if x|a and x|y, then x | s*a + t*b.
s*x*y + t*x*z, we can extract x to make x * (s*y + t*z).

Also, s * a + t * b = gcd(a, b). Will be useful in a minute, trust me.
Let d be smallest positive such linear combination of a and b
D = s*a + t*b
A mod d = a – qd
= a – q(s*a + t*b)
= a(1 – qs) + b(-qt)
so a mod d is a linear combination too.
0 <= a mod d < d
so a mod d = 0
so d | a and d | b, so gcd(a,b) >= d
we know gcd(a,b) | d (gcd(a,b) divides a and b, and d too, since d is lin comb of a,b
and we know d > 0, so gcd(a,b) <= d
this gives us gcd(a,b) = d

So, Extended Euclidean Algorithm is necessary, keep track of quotients as we go

Step through algorithm: pg 26. Write up in python?

```
def eegcd(a, b):
    if b == 0:
        return a, 1, 0
    else:
        rp, sp, tp = eegcd(b, a % b)
        return rp, tp, sp - (a/b) * tp
```

- Base case: return a, 1, 0, so a = as + bt
- Recursive case:
  - Compute rp = b*sp + (a – [a/b] * b)*tp
  - Since r = rp, rewrite this as
  - r = a * tp + b(sp – [a/b] * tp)
    - by distributing tp and then refactoring

Mod
- We all know how it works. R is residue or remainder

Residues
- We create the set of residues for a number n, call them Zn
- All numbers >= 0 and < n

Congruence
- Maping from Z to Zn is not one-to-one, but many to one.
- 2 mod 10 is 2, 12 mod 10 is 2, etc
- congruence is a tripe =, many to one mapping.

Circular notation
- Mod is like a circle, or a clock, wrapping around every 12 hours

Operations in Zn, use mod to make it all work
- 7 + 14 in Z15. Do 7 + 14 = 21, then mod 15 to get 6
- we can always do mods first, to prevent numbers from being huge.
- (a + b) mod n = [(a mod n) + (b mod n)] mod n
- (a – b) mod n = [(a mod n) – (b mod n)] mod n
- (a x b) mod n = [(a mod n) * (b mod n)] mod n
- and 10^x mod n = (10 mod n) ^ x mod n
- 6371 mod 3 = (6 + 3 + 7 + 1) mod 3
  - why?
  - From previous rules, push in the mod, then 10 mod 3 = 1, so it reduces to 1 in them all

Inverses, hooray, this is what we're looking for
- We want to encrypt a message, then be able to get it back to where it was.
- Since mod world is circular, we only have to go forward, and it will take us back,
  - No reverse with – or division
- Additive
  - Identity is 0, a + a-inverse+ = 0
  - Normally, inverse is –n for all n.
  - In Z26, inverse is 26 – a. This means a + b == 0 (mod n) congruent
    - It's all within the set
- Multiplicative
  - Identity is 1, a * a-inverse* is 1
  - Normally, multiplicative inverse requires rational numbers. 5 * (1/5) = 1
  - In Zn, a number has a multiplicative inverse if gcd(a, n) === 1
    - So there's no way for it to keep skipping over 1

It’s relatively prime

Use eegcd to find multiplicative inverse of a in Zn

Find gcd(n, a), and save t, (WHY?)

We know s * n + a * t = 1, because of eegcd

From additive mod, we can do mod on the first part, n*s mod n = 0

So we just care about the second part, a * t === 1

Ok, so we have Z26. is there a multiplicative inverse of 15?

They are relatively prime, with gcd = 1, so yes.

What is it? Use eegcd to find = 7

15 * 7 = 105, and 105 % 26 = 1, right back where you started.

In different sets, some # have mult inverses, some don’t.

When n = a prime number, then all numbers have an inverse except 0, which we don’t really want anyway, since you haven’t done anything to the data.

Next time, Dr. Schlatter talks about Caesar, affine and monoalphabetic ciphers.

Lecture 5: Polyalphabetic ciphers

Cs.centenary bookmarks, this week, Brent
Tag them with your name so I know who added what.

Quiz: Decrypt EPPEMS EP DEOR
You break into Alice's computer, and encrypt LO into JI before you're booted
Affine cipher When you know LO goes to JI
What type of attack is this?

Keys, mult is 17, add is 4. This will take a while, *inverse of 17 is 23

http://xkcd.com/177/ Eve

Discuss Little Brother

> Agreements?
> Disagreements?
> Hacking the rfid cards on cars and such, ethical? Using innocents?
> Issues from our class so far, confidentiality, hiding traffic analysis?

http://javabat.com/ for extra credit.

Repetions of key process each time. This is bad

Common way to make a substitution cipher? Use a keyword, remove all repeated letters, then you only transmit this secretly. Map first letters to these, then fill in the rest. Try to include a y in your keyword, because otherwise it sticks out, vwxz not common, ok to have them map to themselves.

You can use this knowledge in a pattern attack, you know the mapping might have a key, see if you can infer what that key is.

We want a stream of different keys. Before, it was all the same for each letter, transformation of one gave us the same every time. Foiled by Statistical Attacks all over.

Autokey. Use the Caesar cipher idea, but Each letter is the next key.
Frequency of letters is now different. How different?

Load up gettysburg.txt
Transform into just letters caps

Write frequency calculation code: from last year in CSC 104

```
Def freq(s):
        Fr = {}
```

```
    For c in s:
        If c in fr:
            Fr[c] += 1
        Else:
            Fr[c] = 1
    Return fr
```

Freq for digrams?

MAKE A FREQUENCY DIAGRAM. Is it different? Yes, no super-prominent E. But, how many keys are there? Only 25, just care about start character.

Write autokey encipherment program

Based on Caesar cipher, write this first

Also, your key stream has English words as the keys.

Lecture 6: Vigenere Crypanalysis

Cyber Symposium class meeting
Register now, either Tuesday or Thursday we will go as a class.

Vigenere is next. We need different keys, but not just based on one starting number.

Use a keyword, and do Caesar cipher repeatedly with a letter from the keyword

Really, a large 2d table for each pairing. Called Tabula Recta. But we treat it like a stream of data.

How to do cryptanalysis on Vigenere cipher? Kasiski method. Find the repeating period, look for common substrings

Then line up based on found keylength, and do Caesar cipher cracking on each group. Still a pattern.

Friedman test

Index of coincidence

Introduction

- Index of Coincidence (IC) is a statistical measure of text which distinguishes text encrypted with a substitution cipher from plain text.
- IC was introduced by William Friedman in *The Index of Coincidence and its Applications in Cryptography (1920)*
- It has been called "the most important single publication in cryptology".

The Idea

- IC is defined to be the probability that two randomly selected letters will be identical.

- Imagine a hat filled with the 26 letters of the alphabet. The chance of pulling out an A is $^1/_{26}$.
- If we had two such hats. The probability of pulling out two As simultaneously is $(^1/_{26})*(^1/_{26})$.
- The chance of drawing any pair of letters is

$$26*(^1/_{26})*(^1/_{26}) = (^1/_{26}) = 0.0385$$

- So the IC of an evenly distributed set of letters is 0.0385

The Idea (cont.)

- Suppose we fill the hats with 100 letters, and had the number of each letter correspond to the average frequency of that letter in the English language.

  (i.e. 8 As, 3 Cs, 13 Es, etc.)

- The chance of drawing any pair of identical letters is $(^{8}/_{100})*(^{7}/_{99}) + (^{3}/_{100})(^{2}/_{99}) + (^{13}/_{100})(^{12}/_{99}) + \ldots = 0.0667$
- This is the IC for English.
- Every language has such an IC, for example:
- Russian: 0.0529
- German: 0.0762
- Spanish: 0.0775

Calculating the IC

The formula used to calculate IC:

$$\frac{\Sigma(f_i * (f_i-1))}{N(N-1)}$$

where $0 \geq i \geq 25$,

f_i is the frequency of the i^{th} letter of the alphabet in the sample,

and N is the number of letters in the sample

Example

The IC of the text THE INDEX OF COINCIDENCE would be given by:

c(3*2)+ d(2*1)+ e(4*3)+ f(1*0)+ h(1*0)+ i(3*2)+ n(3*2)+ o(2*1)+ t(1*0)+ x(1*0) = **34**

divided by N*(N-1) = 21*20 = **420**

which gives us an IC of 34/420 = **0.0809**

The IC of the text BMQVSZFPJTCSSWGWVJLIO would be given by:

b(1*0)+ c(1*0)+ f(1*0)+ g(1*0)+ i(1*0)+ j(2*1)+ l(1*0)+ m(1*0)+ o(1*0) + p(1*0)+ q(1*0)+ s(3*2)+ t(1*0)+ v(2*1)+ w(2*1)+ z(1*0) **= 12**

divided by N*(N-1) = 21*20 **= 420**

which gives us an IC of **12/420 = 0.0286**

How is this helpful?

- IC can be used to test if text is plain text or cipher text.
- Text encrypted with a transposition cipher would have an IC closer to 0.0385, since the frequencies would be closer to random.
- Substitution cipher will have IC closer to 0.0667 because frequencies will not change.
- English plaintext would have an IC closer to 0.0667.
- This measure allows computers to score possible decryptions effectively.

Then rearrange to get them all worked out.

Use Chi 2 test

(Observed – expected) ^ 2 / expected. Sum for all 26 letters.

They need: Do in class, in python, save in data directory for lab.

Caesar encrypt/decrypt
Affine encrypt/decrypt
Autokey encrypt/decrypt
Frequency Count generator

Lecture 7: Solitaire and Playfair

Register for Cyber Summit

Project Topic Selection

WEP/WPA
Digital Signatures
Authentication/passwd
SSL/TLS
IPSec
Key Management Kerberos
PGP
SSH
AES
Twofish
Database Security k-anonimity
MD-5/Birthday attack
VPN

Shift from stream to block ciphers

Not just one letter at a time, but multiple letters all at once become groups of multiple letters

Smallest is Playfair. Make a 2d array of your letters based on keyword.
"GROUNDHOG"

G R O U N
D H A B C
E F I K L
M P Q S T
V W X Y Z

Encryption: Take message two letters at a time. and replace repeated letters with X in between. If odd in length, tack X on the end.

BOOKKEEPER becomes BO OK KE EP ER
But ABOOKKEEPERSLEEPS becomes AB OX OK KE EP ER SL EX EP SX

Now look up pairs in table.

If they are in same row, replace with letter immediately to the right (mod to wrap around)

If they are in same column, replace with letters immediately below

Otherwise, replace with letters at corners of box formed. First encrypted is at same rwo as first plaintext letter

BO -> AU
OK -> UI
KE -> LF
EP -> FM
ER -> FG

AUUILFFMFG

Not the best, tries to be polyalphabetic, but not much.

Try ABOOKKEEPERSLEEPS

How to cryptanalyze? Look for digrams, frequency of those will be preserved. Find ER RE, TH HE, make guesses based on these letters and see what happens.

Also, use hillclimbing with random matrix. Evaluation metric? IoC, frequency of digrams, matches in English dictionary, etc.

Last time, one time pad discussion. How to do this without using a Geiger counter?

Pseudo-random number generation

LCG are bad, they repeat and are predictable
X_n = a * x_n-1 + b (mod m)

SHA, or BBS (Blum, Blum, Shub) are much better, won't be described here.

Solitaire cipher. Pseduo-random numbers in the palm of your hand.

1. A joker down 1
2. B joker down 2
3. Triple cut (jokers and cards inbetween do not move)
4. Count cut (bottom card, convert to number. Jokers 53. cut after the card you count down to, leave bottom card unchanged)
5. Find output number. Count down using top card. (do not modify deck)

If joker, do nothing and start over.

Decrypt the following

AKSDM INBBW

Key is KASISKI. Use key to set up deck. Replace step 5 with another count cut based on letter in passphrase.

2S 3S 4S 5S 3C 1J 10H 11H 2H 3H 11D 12D 2J 6S 7S 8S 9S 10S 11S 12S 13S 5C 6C 7C 8C 9C 10C 11C 12C 1C 2D 3D 4D 5D 6D 7D 8D 9D 10D 1D 13D 1H 13C 4H 5H 6H 7H 8H 9H 2C 12H 13H 1S 4C

9D
10H
12D
3C
6D
12S
12S
10S
9H
12S

Not completely random though, IoC = 0.0444, so there's some bias.

USE LARGE KEYS, USE SMALL MESSAGES

Lecture 8 Transposition

What can we do for cryptanalysis so far?
- Brute force
- Look for frequency count
  - Use chi squared to see if Caesar shift
  - Frequency, Digrams and trigrams for substitution
  - Kasiski test for vigenere
  - Digrams for playfair cipher

But what if frequency of letters comes back perfect, but digrams and trigrams are disturbed?

Transposition cipher, rearrangement, anagram

Famous anagrams

Seen alive? Sorry, pal! Elvis Aaron Presley
Old West action   Clint Eastwood
Select odd words, laughing. Charles Lutwidge Dodgson

Summary anagrams are anagrams of quoted passages from literature that convey the essence of the work itself. This style is a favorite genre of anagrammatists such as Simon Woodard. Below is an example of one of Woodard's polished summary anagrams, of the first lines of a popular translation of Homer's Odyssey:[3]

"Sing to me of the man, Muse, the man of twists and turns, driven time and again off course, once he had plundered the hallowed heights of Troy." – Homer's Odyssey

Summary anagram:

Hurrying home to his wife, Odysseus shoved off, fled the sea god's wrath, endured many moments of mistreatment, then landed on southern Ithaca... a long epic!

Rail cipher

My Dog Has Fleas

Column Cipher
Column cipher with key for swapping columns=
Column Cipher repeated

How to decrypt? Look for patterns of vowels, this should not be there. Look for separation of letters in common digrams, they could give indication to pattern. Perform multiple anagramming. GHINT and OWLCN. Draw complete graph.

Next, bifid, trifid. Rearranging the bits.

Lecture 9: Enigma Machine and rotors

Choose topic by Thursday, tell me top 3, I let you know what you get.

http://www.youtube.com/watch?v=DnBsndE1IkA

pocket enigma

http://enigmaco.de/enigma/enigma.html

As hard as substitution 26! For brute force
Also polyalphabetic cipher

Plugboard made things complicated (steckerbrett in german)

Why broken?

Reflector was "cryptologic disaster", no letter could be encrypted to itself.

Human errors

Rotor setup, repeat three letter sequence twice at beginning of message.

One person wrote frequently "NOTHING TO REPORT"

Numbers translated into letters, looked for "EINS" German for 1.

Used Crib technique, looked for frequent words, see if this decrypts the rest.

"Gardening" to seed the messages. Chosen Plaintext attack.

Do encryption/Decryption with paper enigma.

WBNDNIGJFDPSMZINXESA, code book says use I, II, III, start at MCK

Lecture 10: Algebraic Structures

QUIZ: Name three encryption algorithms we've discussed. List:
    Block or stream
    Keysize
    Frequency preservation? At what level?

We have been working in Z26 a lot with our ciphers so far.

But we are in the computer age, let's start working with binary data., 2^n

We still want the same things to work though, +, -, *, /, but on bits

Groups, Rings and Fields

GROUP

A group has one binary operation, #. Need properties on this operation

    Closure a, b then a # b = c is element of G
    Associativity (a # b) # c = a # (b # c)
    Existence of Identity e # a = a # e = a
    Existence of Inverse a # a' = a' # a = e

In commutative or abelian group,
    Commutativity a # b = b # a

Zn with + is an abelian group. Everything is satisfied.

Zn* with *, and only elements with inverses, is an abelian group.

Permutation group, set is of all permutations, operation is composition.

EXAMPLE HERE

Letters 1, 2, 3 permutations as [1, 3, 2], [2, 3, 1], etc…

Show with transformation pictures, getting ready for S/P Boxes

Non-abelian, commutative does not work. PROVE THIS HERE

RINGS

Two operations, one is abelian, the second must have CLOSURE, ASSOCIATIVITY
Identity of first operation has no inverse.

Z with + and * is a commutative ring. Division yields an element out of the set

Will be used for Prime numbers in the future…

FIELDS

Commutative ring with all five properties for the second operation.
We want **Finite Fields**, dealing with bits, not floating point numbers here
Number of elements should be p^n, called Galois fields, so

GF(2^n) is the Galois field with 2 as our prime number of choice, (for bits)

Look at GF(p) first, so GF(2)

Elements 0, 1, operations + *

Table for addition (really Exclusive Or XOR) plus with circle

Table for multiplication

GF(2)

{0, 1} + ×

+	0	1
0	0	1
1	1	0

Addition

×	0	1
0	0	0
1	0	1

Multiplication

a	0	0
$-a$	1	1

a	0	1
a^{-1}	—	1

Inverses

Inverses

Also works for GF(5). This is the same as Z5. But we want bits. 2^n, and these will not be prime numbers, not all elements will have a multiplicative inverse. ☹ use fewer elements, not really good. Define new operations on the numbers. BETTER.

GF(5)

{0, 1, 2, 3, 4} + ×

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Addition

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Multiplication

Additive inverse

a	0	1	2	3	4
$-a$	0	4	3	2	1

a	0	1	2	3	4
a^{-1}	—	1	3	2	4

Multiplicative inverse

Specific. GF(4) can we make it work?

(+) table from book

(*) table from book

Addition

⊕	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

Identity: 00

Multiplication

⊗	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

Identity: 01

What's the theory behind these?

POLYNOMIALS

Remember how we write numbers can be expanded

1729 is 1 * 10^3 + 7 * 10^2 + 2 * 10^1 + 9 * 10^0

and binary numbers use 2^n instead of 10^n, with the coefficients being 0 and 1 only.

n-bit word	1 0 0 1 1 0 0 1
Polynomial	$1x^7 + 0x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 0x^1 + 1x^0$
First simplification	$1x^7 + 1x^4 + 1x^3 + 1x^0$
Second simplification	$x^7 + x^4 + x^3 + 1$

Our theory will be about these polynomials, use x^n instead of 2^n so we can deal with the mathematics in an abstract way, since we really just care about the coefficients, but need the polynomials underneath them to rearrange their theory.

Addition. Use XOR on the coefficients. Always stay within the bounds of the finite field

Additive identity: zero polynomial

Additive inverse: yourself. XOR with yourself gives you 0 everywhere.
Subtraction exactly the same. GOOD FOR CRYPTOGRAPHY

Multiplication.. it looks like we'll go outside of our bounds for the finite field, get elements beyond our ken.

But just like with Z26, we can do MOD, but it's much more complicated now.

Multiply like normal. 00100110 * 10011110.

(X5 + x2 + x) * (x7 + x4 + x3 + x2 + x)

distribute. Drop out duplicates because of XOR of coefficients.

We get x12 + x7 + x2, definitely too big. We need a mod.

Irreducible polynomial of degree n. prime polynomial, cannot be factored.

For degree 2, we have x2 + x + 1. Can't reduce it any further. This is our Modulus.

For 8, we have x8 + x4 + x3 + x + 1.

So, we do polynomial long division.

X12 + x7 + x2 / x8 + x4 + x3 + x + 1.

Quotient is x4 + 1.

Remainder is x5 + x3 + x2 + x + 1.

$$
\begin{array}{r|l}
 & x^4 + 1 \\
\hline
x^8 + x^4 + x^3 + x + 1 & x^{12} + x^7 + x^2 \\
 & \underline{x^{12} + x^8 + x^7 + x^5 + x^4} \\
 & x^8 + x^5 + x^4 + x^2 \\
 & \underline{x^8 + x^4 + x^3 + x + 1} \\
\text{Remainder} & \boxed{x^5 + x^3 + x^2 + x + 1}
\end{array}
$$

Multiplicative Identity is still 1

Multiplicative Inverse??? EEGCD generalized…. Ugh. We’ll trust that to work…

But we can make multiplication much easier on ourselves, no long polynomial division, hooray!

Say we have polynomial. P. x3 * P = x * x2 * P = x * x * x * P

We can build this up incrementally.

Same problem as before.

Go from x0 to x5. Need to do reduction twice. DO THIS IN CLASS
When you multiply by x, you are just shifting the bits. When you get too big, you are XORing the bits with the irreducible polynomial.

In computers, these are very easy operations!! ^ does XOR, << does bit shifting, in python, same thing in java.

Lecture 11: Elements of Modern Block Ciphers

Next Week Tuesday: Project Source Summary. 3 sources, discussion of each and how they will help with your presentation

Where are we going next?

Modern Block Ciphers
Simple DES
Practical Use
Prime Numbers
RSA
Defense against the Dark Arts
Guest Lecture?

Two basic things to take away from classical ciphers

Transposition and substitution.

Both are used in modern ciphers in different places

Substitution has much larger transformation space, for 64 bits, it's 2^64 no matter what the message

Transposition, smaller, 64! / (10!) (54!) if there's 10 1s in message
151,473,214,816

How to do these with a block cipher on bits?

Transposition is easy, just use permutation group

3 bits come in, permute, 3 bits go out

Substitution is more difficult, but possible.

3 bits come in, expand to 8 bits, with 7 0s and 1 1s.
permute the expanded bits
compress the permuted 8 bits back into 3. This is your substitution

How big is your key, to say what you permute?

Transposition, need 3! = 6 elements, floor(log2(6)) = 3 bits
Substitution, need 8! = 40,320 elements, floor(log2(40,320)) = 16 bits.

More than one stage is useless. (Permutation reduction) And if we want blocks of 64 bits, we need enormous keys to decide which permutation to use.

So we need partial-size key ciphers. Break the exact permutation group property, and let us have smaller keys

Components of Modern Block Cipher

P-Box

Transposes the bits, p for permutation. Can be fixed (keyless) or keyed, dependent on the outside. Straight P-Box is n to n. it's invertible

Compression P-Box leaves some bits out. Not invertible

Expansion P-Box reuses some of the bits. Not invertible

Why do we care? They can be used in other ways and cancel the effect.

S-Box

Miniature substitution cipher. N to m possible. Is invertible if n to n. Usually predefined, not dependent on the key. Can be linear formula y1 = .. y2 = …

Non-linear formula, $y1 = (x1)^3 + x2$, $y2 = (x1)^2 + x1x2 + x3$

Or just a table definition mapping inputs to outputs.

Exclusive Or

All those nice properties we talked about last time. Inverse works if you have fixed data for one of the inputs.

Circular shifting

Shift bits, and wrap around. Left is inverse of right, vice versa

Shift is modulo n. This is like permutation group, doing it more than once does nothing.

Swap

Left/Right shift of n/2. This is self-invertible

Split and combine

Split bits in the middle to make two equal-length words.

Combine is the inverse, take two equal-length words and concatenate them.

Product Ciphers

Introduced by Shannon, Major player in information theory

Combination of Substitution, Permutation and other components.

Diffusion and Confusion

Diffusion- Hide relationship between plaintext and ciphertext (1 different bit in plaintext changes many in ciphertext)

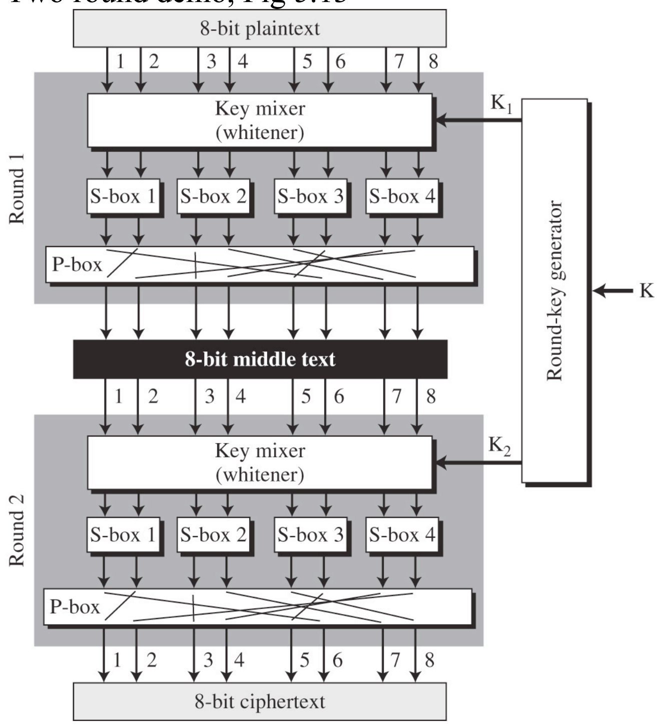
Confusion – Hide relationship between ciphertext and key (1 different bit in key changes many in ciphertext)

Rounds

Iterate this process. Use a Key Generator or Key Schedule to make different keys each time.

Lecture 12: Our Friend Feistel

Two round demo, Fig 5.13

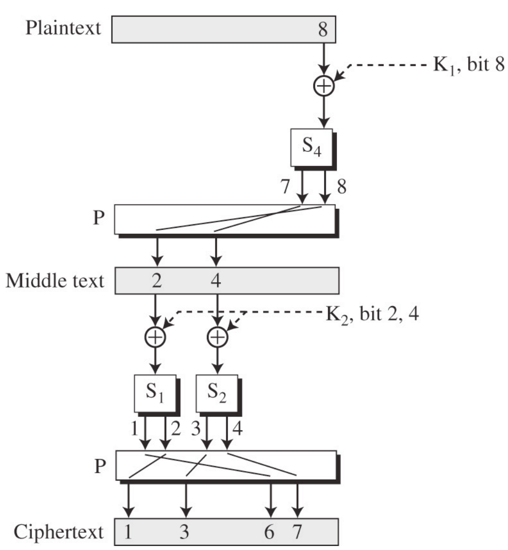


Input text mixed XOR with key (called Whitening)

Outputs organized into four two-bit groups. Fed into four s-Boxes.

Outputs fed through permutation, makes it different next round.

Figure 5.14 shows the diffusion and confusion. ADD MORE rounds, more diffusion and confusion

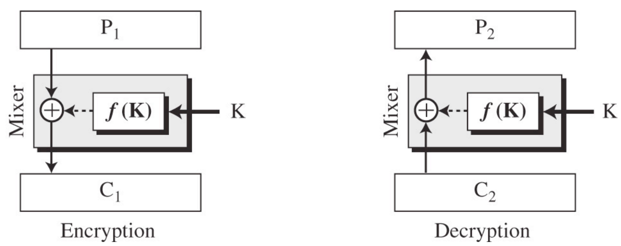


Read about Feistel Ciphers

We can have the decryption algorithm exactly the same at encryption. ROT13 is example of this. Affine, we needed different algorithm. Ok, if you work with software, but not with hardware, you want to make minimum number of circuits.

Feistel found a way to have bit-wise encryption be self-invertible, and still have confusion and diffusion.

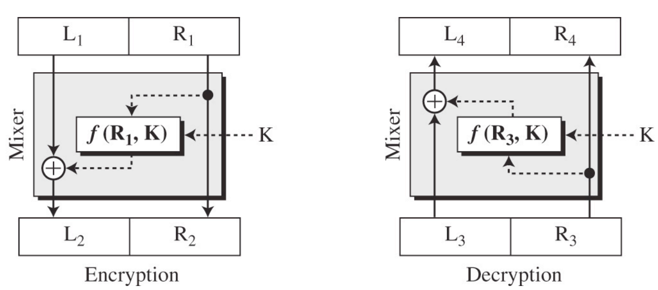
Proposal 1:



f(K) can be non-invertible, can be anything we want. Needs to be non-linear in some way.

Prove that P1 = P2, assuming C1 = C2.
P2 = C2 XOR f(K)
= C1 XOR f(K)
= P1 XOR f(K) XOR f(K)
= P1

Proposal 2:



But this is only reliant on the key, not anything in the plaintext. Not good diffusion of information.

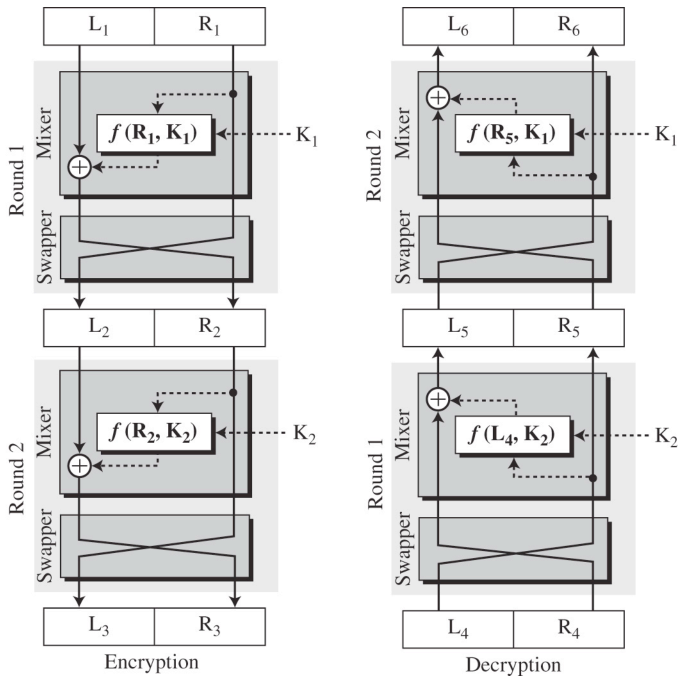
Split into L1 and R1, use R1 as part of the function.

Prove L1 = L4 and R1 = R4, if L2 = L3 and R2 = R3

R1 = R2 = R3 = R4

L4 = L3 XOR f(R3,K)
  = L2 XOR f(R3, K)
  = L2 XOR f(R2, K)
  = L2 XOR f(R1, K)
  = L1 XOR f(R1, K) XOR f(R1, K)
  = L1

Proposal 3: TYPO, CHANGE R5 to L5 in upper right box



R1 was part of function, but it never changed, R4 = R3 = R2 = R1

Introduce swap, so each round, left becomes right, and right becomes left.
Need to do this more than once. Need different key each time too. Make key schedule…

Can we still decrypt with same algorithm, but reversed key schedule?

Proof that L1 = L6 and R1 = R6, given L3 = L4 and R3 = R4

R5 = L4 = L4 = R2
L5 = R4 XOR f(L4, K2)
  = R3 XOR f(L4, K2)
  = R3 XOR f(L3, K2)
  = R3 XOR f(R2, K2)
  = L2 XOR f(R2, K2) XOR f(R2, K2)
  = L2

so middles are equal

R6 = L5 = L2 = R1

L6 = R5 XOR f(L5, K1)
  = R2 XOR f(L5, K1)
  = R2 XOR f(L2, K1)
  = R2 XOR f(R1, K1)
  = L1 XOR f(R1, K1) XOR f(R1, K1)
  = L1

Repeat many times, and you are set with confusion and diffusion.

Leave out the last swap, and encryption identical to decryption.

Non-Feistel

Differential Cryptanalysis
Linear Cryptanalysis

Worry about Attacks Later.

Read Appendix O, pg 659 of the book

Lecture 13: SDES Runthrough

Example in Book is Incorrect ☹

Lecture 14 : Attacks on Block Ciphers

SDES Example in book is wrong, correct output is 01110111

For left part of SBox, use 1, then 4, not following the arrows.

Differential Cryptanalysis

Find the cipher key.

Chosen plaintext

Examine simple cipher in 5.19 figure

$X1 + X2 = P1 + P2$,

If we just had xor as our cipher, then it's easy to find this relationship, don't even need the key.

SBox prevents us from finding a definite relationship.

But we can get a probabilistic relationship… Set up Differential Distribution Table for each SBox in your cipher.

Linear Cryptanalysis

Known plaintext

Lecture 15: Modes of Operation

Block ciphers of DES, AES, etc. Work on 64, 128 bits of data at a time.

How to use this as an actual cipher for very long messages?

Modes of Operation

Electronic Codebook (ECB)

Sequentially run through the data, encrypting every 64 bits with the same key.

Good news:
    Errors are not propagated in the whole message
    Algorithm is simple
    Can be easily parallelized

Bad news:
    Patterns at block level preserved.
    Replay attacks.
    Show linux penguin on wikipedia

Cipher Block Chaining (CBC)

Before enciphering, do an XOR with the output of the previous block

What about first block?

Need IV (Initialization Vector). How to transmit, as key, enciphered as one block?

Good:
    No longer have repeated blocks from same plaintext.
    Very common encription use method
    Errors in ciphertext still do not propagate, contained to one block…?? WHY?

Bad:
    If first M blocks are same in two messages with same key, ciphertext will be the same
    Eve can tack on extra information to the end of the message.
    Errors in plaintext propagate to all future

Ciphertext Stealing. Not as bad as it sounds. If you have message with length not multiple of 64, you need padding. Random numbers? Instead, reuse portions of the code.

Encipher N-1 block. Use tail of block as part of N block and encipher again.

Cipher Feedback Mode

Used for encrypting smaller pieces of information, such as 8 bit ASCII characters.

Set up a shift register. Use IV again. For each new subset of bits, shift the resulting cipher into the next portion.

Output Feedback Mode

Ciphertext is independent of previous bits

Lecture 16

Symmetric key – the key is secret and transmitted by some other secure channel.

Used for large messages.
Key is a string of bits
With N people, symmetric needs (n * (n – 1)) / 2 keys.
Can be very fast

Asymmetric. Keys (more than one) are both public and private. All information is not known by all parties, only enough to do the encryption.

Used for small messages. Exchange of keys for symmetric key messages.
Keys are numbers
With N people, asymmetric needs 2 * n (public, private)
Usually slow computationally

One for locking, the other for unlocking

Analogy of Post Office:

Alice: Bob, send me your padlock.
Bob sends padlock
Alice writes message, locks with Bob's padlock.
Bob gets message, opens with private key.

Mathematically:

Ciphertext = f(K_public, Plaintext)
Plaintext = g(K_private, Ciphertext)

We want g to be similar to f^-1 inverse, but with special properties

1. f(x) is easy to compute
2. f -1(x) is difficult to compute
3. given y and trapdoor, x is easy to compute. (our function g)

Key security. How do you know that the public key Bob posted is really Bob's key and not Eve's?

Certificate Authority. This is the message you get when you try to log into Wireless at Centenary. Our Key is not Authenticated (Costs money, hassle).

Let's see how this might be put into practice.

Knapsack cryptosystem. Based on subset-sum problem. In general, NP Complete problem.

Here are 5 items I own, and how much they weigh.

2.3
3.7
4.3
3.8
5.1

I tell you my backpack weighs 7.4. What items am I carrying? 1 and 5.

This problem is hard, you need to evaluate every subset, calculate the sum, and there's lots of possible subsets. 2 ^ N

In general, you have the subset you want as the message you want to send. This is your number, and the binary representation of that number lets you denote 1 for "add item" and 0 for "do not add item". It takes Eve 2^N choices to find your subset choice, how can this be easier for Bob?

There are some sets where the subset sum is very easy to find without exponential search.

What is binary representation of 34?

100010

You just solved a subset-sum problem. The key was that the items were a superincreasing tuple.

a_i >= a_1 + a_2 + … + a_i-1

1, 2, 4, 8, 16 fits this property, and any subset can be quickly found in linear time.

How does this help us?

We keep a superincreasing tuple as a private key, but release it to the world as a public key with some small modifications with mod.

1. Create superincreasing tuple

1, 2, 4, 8, 16, 32, 64, 128

2. Choose modulus n so that n > sum of everything before. So N > 256. We choose 359.

3. Choose r that is relatively prime with n and 1 <= r <= n -1
    since 359 is prime, then any number in the range will work. Pick 55

4. Create new tuple, where each item is r * b_i mod n
[55, 110, 220, 81, 162, 324, 289, 219]

5. Permute these to a new arrangement. 4, 2, 8, 5, 3, 6, 1, 7

[81, 110, 219, 162, 220, 324, 55, 289]

This is the public key, r, n and original tuple set are the private key.

1. Alice wants to send a message, the letter M. this is number 77, so we have binary 01001101.

2. She calculates the sum, 110 + 220 + 324 + 289, 943, and sends this to Bob.

Bob now has the message, private key time.

r-1 is 235, from eegcd alg.

1 calculate s' = r-inverse * s mod n

235 * 943 % 359 is 102

calculate inverse knapsack (for us, binary of 102) = 01100110.

Permute this as above.   01001101. This is 77, whala, our magic text!

How did this work? MAGIC.. no, mathematics.

Each number in the tuple is r * b_i mod n. Add them together you get

r * b_2 + r * b_3 + r * b_6 + r * b_7.

Multiply by r-inverse, mod n, it wipes out all the rs, leaving us with b_2 + b_3 + b_6 + b_7.

This is from the superincreasing tuple, so we can find which elements. 2, 3, 6, 7.

Then we need to know where they were in the public tuple, so we apply forward permutation again, to get the message x.

This was unique when developed.

Cryptanalysis??

Lecture 17

We saw simple knapsack last time. Quiz.

Bob's private key is b = [1,2,4,8,16,32,64,128], n=271, r= 87

Bob's public key? [87, 174, 77, 154, 37, 74, 148, 25]

Alice sends the message 322 to Bob. Decrypt this message.

B

We need larger system. Move to using Prime numbers

Prime number review

Infinite number of primes. Proof from students?

How many primes less than n? has upper, lower bounds, but not exactly defined.

Sieve of Eratosthenes. Repeatedly cross out from the list multiples as you reach a new number. Only need to go to sqrt of max number in list.

Can do this with lazy computation… later, talk if interested

Totient function: from Euler.

Not how many numbers are prime less than n, how many numbers less than n are **relatively prime** to n.

1. phi(1) = 0
2. phi(p) = p – 1 if p is a prime
3. phi(m x n) = phi(m) x phi(n) if m and n are relatively prime
4. phi(p ^ e) = p ^ e – p ^ (e – 1) if p is a prime

Always breaks down to prime numbers for composites, if we can factor them.

Phi(13) = 12
Phi(10) = phi(2) * phi(5) = 1 * 4 = 4
Phi(240) = phi(2^4 * 3 * 5) = phi(2^4) * phi(3) * phi(5) = 8 * 2 * 4 = 64

In Zn*, phi(n) tells us how many numbers have distinct multiplicative inverses.

Demonstrate for 5, 6, 7, 8
Phi(5) = 4
Phi(6) = phi(2) * phi(3) = 1 * 2 = 2

Phi(7) = 6
Phi(8) = phi(2 ** 3) = 2**3 – 2 ** 2 = 4

Fermat's Little Theorem
a ** (p-1) = 1 mod p
a ** p == a mod p. If p is prime, and a is integer.

Quickly find (3 ** 12) mod 11

= (3 * 3 ** 11) mod 11
= (3 mod 11) * (3 ** 11 mod 11)
= 3 * 3 mod 11
= 9

also, a inverse mod p = a ** (p – 2) mod p
multiply both sides by a, you get 1 mod p.

Quick way to find multiplicative inverse in some situations

Euler's Theorem

If a and n coprime, a ** phi(n) == 1 (mod n)

If n = p * q, a < n and k an integer, then a ** (k * phi(n) + 1) == a (mod n)

What? Just trust me. If you want proofs, go to modern algebra.

Quick way to find inverses mod a coprime

If n and a are coprime, a inverse mod n = a ** (phi(n) – 1) mod n

Skip primality testing for now. Assume we have really big prime numbers.

Factorization?

Fundamental Theorem of Arithmetic

Everything can be reduced to primes.

Finding that factorization is hard.

Trial division method.
    Try all positive integers starting with 2, find one that works.

Others are faster, but nothing polynomial, just exponential.

Skip quadratic congruence for now.

Exponentiation and Logarithms..

How do you find exponents?   2 * 6 is 2 * 2 * 2 * 2 * 2 * 2

But is there a faster way? Especially when we have really large exponents?

Fast Exponentiation, the square and multiply method

Exponent can be treated as bits.

So y = a ** x is a ** binary rep of x. if bit is 0, it disappears.

a ** 9 = a ** 8 * a

in general, we can for loop through squares, and multiply them as we go.

Also, works for mod. Always move the mod inside the multiplication, so we can do mod every single step.

So, we can find exponential number in polynomial time.

How about the reverse. If we encrypt with exponentials, can we decrypt quickly with logarithms?

y = a ** x, then log_a of y = x

Exhaustive search, exponential time, try all exponents.

Discrete logarithms. Need more background in abstract algebra than I have. Nice algorithms, but never polynomial, all exponential.

Lecture 18: RSA

C = P ** e mod n

E and n are public key

P = C ** d mod n

D and n are private key

How does this work?

First, choose two very large prime numbers, p and q.

Calculate n = p * q

Find phi(n). = phi(p) * phi(q) = (p – 1) * (q – 1)

Select e such that e coprime to phi(n). and 1 < e < phi(n)

Find d = e inverse mod phi(n). Fast because Euler said *a inverse mod n = a ** (phi(n) – 1)*

Encryption fast, polynomial time.
Decryption without key slow, exponential time to solve RSA problem.

Decryption with key fast, polynomial time. Why does it work?

> C ** d mod n = (P ** e mod n) ** d mod n = P ** ed mod n
>
> ed = k * phi(n) + 1, since ed is 1 mod phi(n).
>
> so rewrite as P ** (k * phi(n) + 1) mod n.
>
> again from Euler above*: then a ** (k * phi(n) + 1) == a (mod n)*
>
> So we have just P mod n = P.

Small example. 7, 11, we get 77. Totient is 60. We choose 13 = e, d = 37.

Alice wants to send 5. She computes 5 ** 13 = 26 mod 77.

Bob receives 26. 26 ** 37 = 5 mod 77.

We recovered the plaintext. Alice would have to search a long time.

Others:

Rabin

E and d are fixed. 2 and ½. Public key is just N, private key is p * q.

Find the message from Chinese remainder algorithm

Gives us 4 possible plaintexts, must choose the correct one. If you know what might have been encoded, this is much easier.

ElGamal

Based on discrete logarithm problem

Elliptic Curve Cryptosystems

???

GPG (open source GNU PGP)

gpg --gen-key

find your public key number

gpg --list-keys

gpg --keyserver keyserver.ubuntu.com --send-keys KEYID

Get other keys from keyserver, save to your space.

gpg --import KEYFILE

Sending messages:

gpg -o encrypted_file.gpg --encrypt -r RECKEYID original.file

gpg --decrypt filename.gpg

Lecture 19 - Authentication via password

Logging into your computer, any remote system. They ask for your password.

1) How to make a strong password

Password crackers don't try all possible passwords from aaaaaaaa - zzzzzzzz

don't use words plus appendage (prefix of suffix)

1000 common passwords plus 100 common appendages (1, 4u, abc) can capture 24% of all passwords in use.

More characters the better. They will check for "leet" speak replacements

Don't use your birthday or year.

Take a sentence and turn it into a password.

"This little piggy went to market" might become "tlpWENT2m"

Use symbols too.

2) How to use passwords for authentication

You need to make a website with passwords for people to have personal information stored.

passwords are not stored in a file.

If someone breaks into the system, they would see all the passwords. BAD

So, passwords are encrypted, and the computer stored the encrypted version of the passwords.

To authenticate, you ask for a password, then encrypt it, and compare to the stored password. (also known as a hash).

Python

from crypt import crypt

crypt("hello", "AA")

Ignore second argument for now

Returns encrypted password. Uses DES.

Weaknesses. Only uses first 8 characters, the rest are ignored.

3) How to use rainbow tables to crack encrypted passwords

We have hash function to take us from passwords to encrypted.

We need reverse function. This is hard.

Precompute all possible passwords? Store it in a table? Would take enormous amount of space.

So we create a reduce function. Takes us back to a password, not necessarily the one that created the hash.

Hop back and forth now between hash and passwords. Longer chains mean smaller table size.

Create many many paths. Store the first and last element of the list.

We have a certain hash with an unknown plaintext, and we want to check to see whether it is inside any of the generated chains.

The algorithm is:

* Look for the hash in the list of final hashes, if it is there break out of the loop.
* If it isn't there reduce the hash into another plaintext, and hash the new plaintext.
* Goto the start.
* If the hash matches one of the final hashes, the chain for which the hash matches the final hash contains the original hash.

You can now get that chain's starting plaintext, and start hashing and reducing it, until you come to the known hash along with its secret plaintext. You know it's somewhere in this chain because you found the end of the chain.

Breaks with the addition of SALT. The second argument on the above crypt function. You'd need to make 4096 rainbow tables. This would be too large.

Lecture 20 - K-anonymity and l-diversity

Hospitals, Census, Google Searches, generate tons of data.

You want to keep data confidential. But, you also want to find trends in the data, do market research, machine learning, statistical analysis.

1 - Easy statistics.

Mean, median, mode. Destroys all individual, into summary information.

For trends and learning, you want some connection to individuals. What should you do?

2 - Remove name, SSN, telephone number, physical address, etc.

Is this enough?

No. 1990 census data summary. From only Zip, gender and birth date, 87% of people identified with information that made them unique. If you knew about someone, you could look them up.

Also, merge across two databased. Health Records, and Voter Registration in Massachusetts.

William Weld was governor of Massachusetts at that time
and his medical records were in the GIC data. Governor Weld lived in
Cambridge Massachusetts. According to the Cambridge Voter list, six people
had his particular birth date; only three of them were men; and, he was the
only one in his 5-digit ZIP code.

Instead use Quasi Identifiers.

7110* for zipcode. 711**.

Birth date, Sep. instead of full date.

Age 3*

Need to balance anonymity with usefulness of the data.

Definition 3. *k*-anonymity
Let RT(*A1*,...,*An*) be a table and *QIRT* be the quasi-identifier associated with it.
RT is said to satisfy *k*-anonymity if and only if each sequence of values in
RT[*QIRT*] appears with at least *k* occurrences in RT[*QIRT*].

You must match K-1 other individuals on the quasi-identifier fields to be K anonymous.

Is this enough?

Homogeneity attack.

You see your neighbor, who you hate, go to the hospital. You get the k-anonymous records. But, it turns out every one of those K people who match your neighbor, all have cancer. So you conclude they have cancer.

Background Knowledge attack

Ok, two diseases as part of your k-anonymous data. But you know your neighbor is Japanese, and they have low heart disease, so they have to have respiratory problems.

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	13053	28	Russian	Heart Disease
2	13068	29	American	Heart Disease
3	13068	21	Japanese	Viral Infection
4	13053	23	American	Viral Infection
5	14853	50	Indian	Cancer
6	14853	55	Russian	Heart Disease
7	14850	47	American	Viral Infection
8	14850	49	American	Viral Infection
9	13053	31	American	Cancer
10	13053	37	Indian	Cancer
11	13068	36	Japanese	Cancer
12	13068	35	American	Cancer

Figure 1. Inpatient Microdata

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	$*$	Heart Disease
2	130**	< 30	$*$	Heart Disease
3	130**	< 30	$*$	Viral Infection
4	130**	< 30	$*$	Viral Infection
5	1485*	≥ 40	$*$	Cancer
6	1485*	≥ 40	$*$	Heart Disease
7	1485*	≥ 40	$*$	Viral Infection
8	1485*	≥ 40	$*$	Viral Infection
9	130**	$3*$	$*$	Cancer
10	130**	$3*$	$*$	Cancer
11	130**	$3*$	$*$	Cancer
12	130**	$3*$	$*$	Cancer

Figure 2. 4-anonymous Inpatient Microdata

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	1305*	≤ 40	$*$	Heart Disease
4	1305*	≤ 40	$*$	Viral Infection
9	1305*	≤ 40	$*$	Cancer
10	1305*	≤ 40	$*$	Cancer
5	1485*	> 40	$*$	Cancer
6	1485*	> 40	$*$	Heart Disease

7 1485* > 40 ∗ Viral Infection
8 1485* > 40 ∗ Viral Infection
2 1306* ≤ 40 ∗ Heart Disease
3 1306* ≤ 40 ∗ Viral Infection
11 1306* ≤ 40 ∗ Cancer
12 1306* ≤ 40 ∗ Cancer

Figure 3. 3-Diverse Inpatient Microdata

Principle 2 (ℓ-Diversity Principle) *A q-block is ℓ-diverse if contains at least ℓ "well-represented" values for the sensitive attribute S . A table is ℓ-diverse if every q$\cdot$ -block is ℓ-diverse.*

T-closeness is next.

l-diversity may be impossible or unnecessary.

What if we have only two results, Pos or Neg on HIV Test, with 99% of results negative. One-sided, if pos, don't identify as pos.

Syntactic diversity, not semantic diversity.

Definition 2 (The t-closeness Principle:) An equivalence class is said to have t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold t. A table is said to have t-closeness if all equivalence classes have t-closeness.