COS433/Math 473: Cryptography

Mark Zhandry

Princeton University

Spring 2017

What is Cryptography?

What is Cryptography

Concise Oxford English Dictionary: *"the art of writing or solving codes"*

Merriam-Webster: *"the enciphering and deciphering of messages in secret code or cipher"*

Wikipedia: *"the practice and study of techniques for secure communication in the presence of third parties called adversaries"*

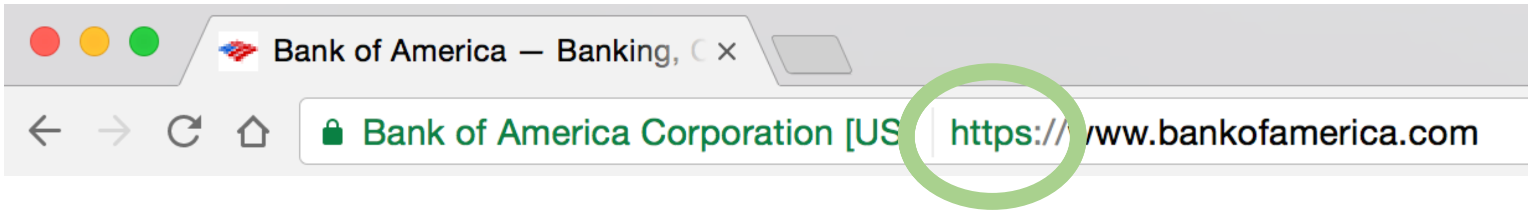
<u>None of these capture the true breadth of the field</u>

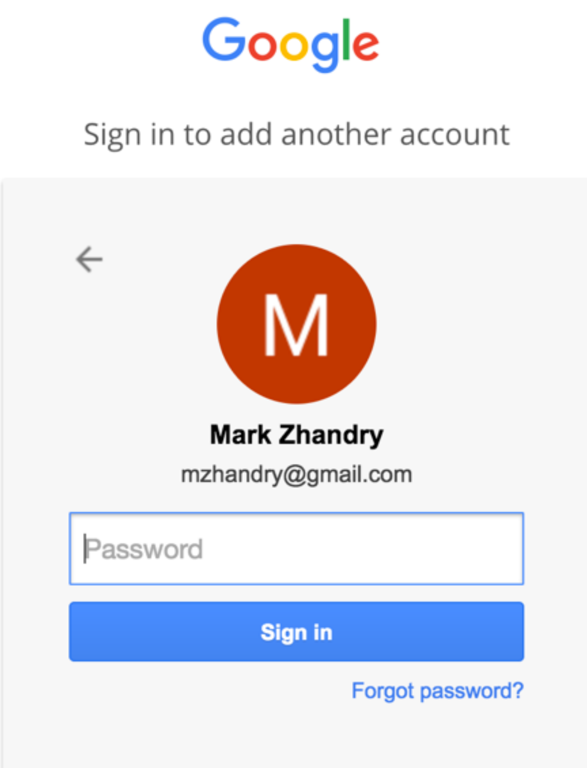
My Definition

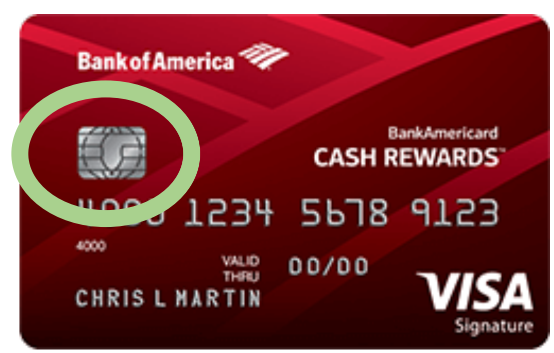
Cryptography is about using secrets to solve interesting tasks

(still doesn’t capture everything)

Cryptography Is Everywhere







A Long & Rich History

Dates back almost 4000 years

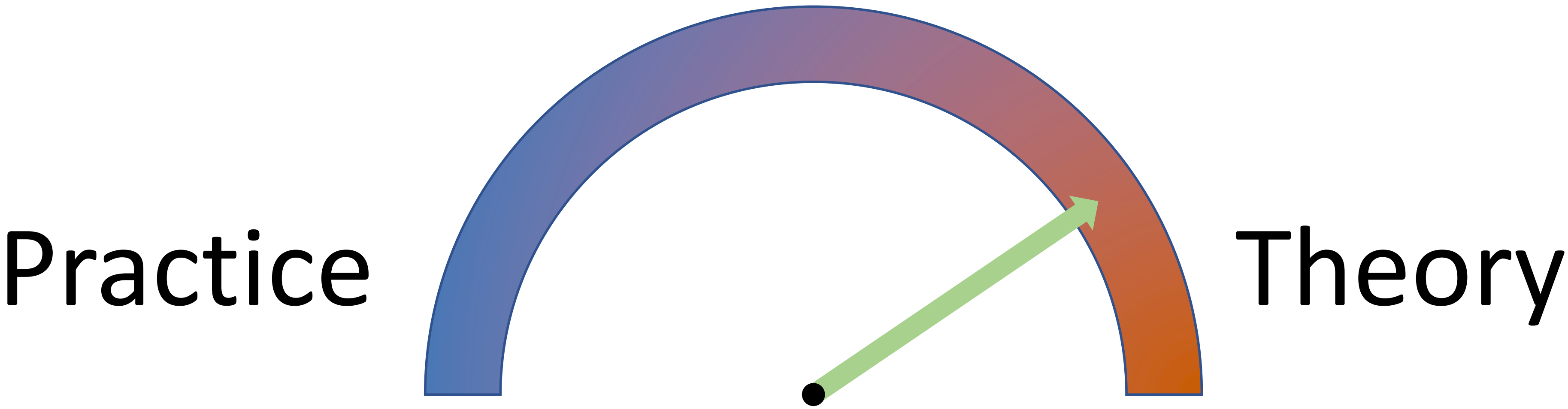
Important historical consequences

- 1587 – Babington Plot
- WWI – Zimmermann Telegram
- WWII – Enigma

Intimately tied to development of modern computer

- First program written for Atlas supercomputer
- First magnetic core memories, high-speed tape drives, all-transistor computers, desktop-sized computers, remote workstations all built based on NSA orders

COS 433



Inherent to the study of crypto

- Working knowledge of fundamentals is crucial
- Cannot discern security by experimentation
- Proofs, reductions, probability are necessary

COS 433

What you should expect to learn:

- Foundations and principles of modern cryptography
- Core building blocks
- Applications

Bonus:

- Debunking some Hollywood crypto
- Better understanding of crypto news

COS 433

What you will **not** learn:

- Hacking
- Implementing crypto
- How to design secure **systems**
- Viruses, worms, buffer overflows, etc

Administrivia

Course Information

Instructor: Mark Zhandry (mzhandry@pr)
TAs: Udaya Ghai (udayaghai@gm)
Qipeng Liu (qipengl@pr)

Lectures: MW 1:30-2:50pm, Friend 008

Webpage: cs.princeton.edu/~mzhandry/2018-Spring-COS433/

Office Hours: please fill out HW0 Doodle poll

Piazza

https://piazza.com/class/jb0zp9b0blf3o0

Main channel of communication

- Course announcements
- Discuss homework problems with other students
- Find project/study groups
- Ask content questions to instructors, other students

Prerequisites

- Ability to read and write mathematical proofs
- Familiarity with algorithms, analyzing running time, proving correctness, O notation
- Basic probability (random variables, expectation)

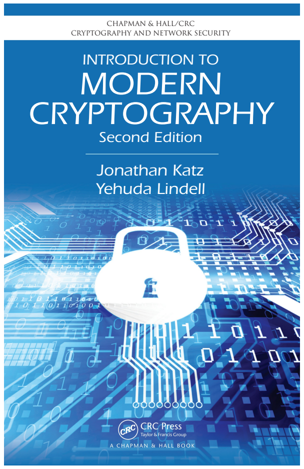
Helpful:

- Familiarity with NP-Completeness, reductions
- Basic number theory (modular arithmetic, etc)

Reading

No required text

But highly recommend:



Introduction to Modern Cryptography
by Katz, Lindell

For each lecture, page numbers for 2nd edition will be posted on course website

Grading

40% Homeworks

- ~1 per week
- No dropped/late hws, but “extra credit”
- Only typed solutions, submitted via CS Dropbox
- Collaboration encouraged, but write up own solutions

30% Projects

- More details a the end of class today

30% Take-home Final

- Individual

Classroom Policies

Please stop me if you have any questions

Please come to class to be engaged and to learn

- Notes for each lecture will be added to the webpage
- I don’t take attendance
- Don’t be on Facebook, working on assignments, etc

Feel free to call me “Mark”, “Professor”, “Hey You”, etc, though “Mark” is preferred

Approximate Course Outline

Week 1: Pre-modern crypto (≤ 1950s)

Weeks 2-6: Foundations of modern cryptography
- Crypto theory
- Symmetric key cryptography

Weeks 7-12: Public key cryptography

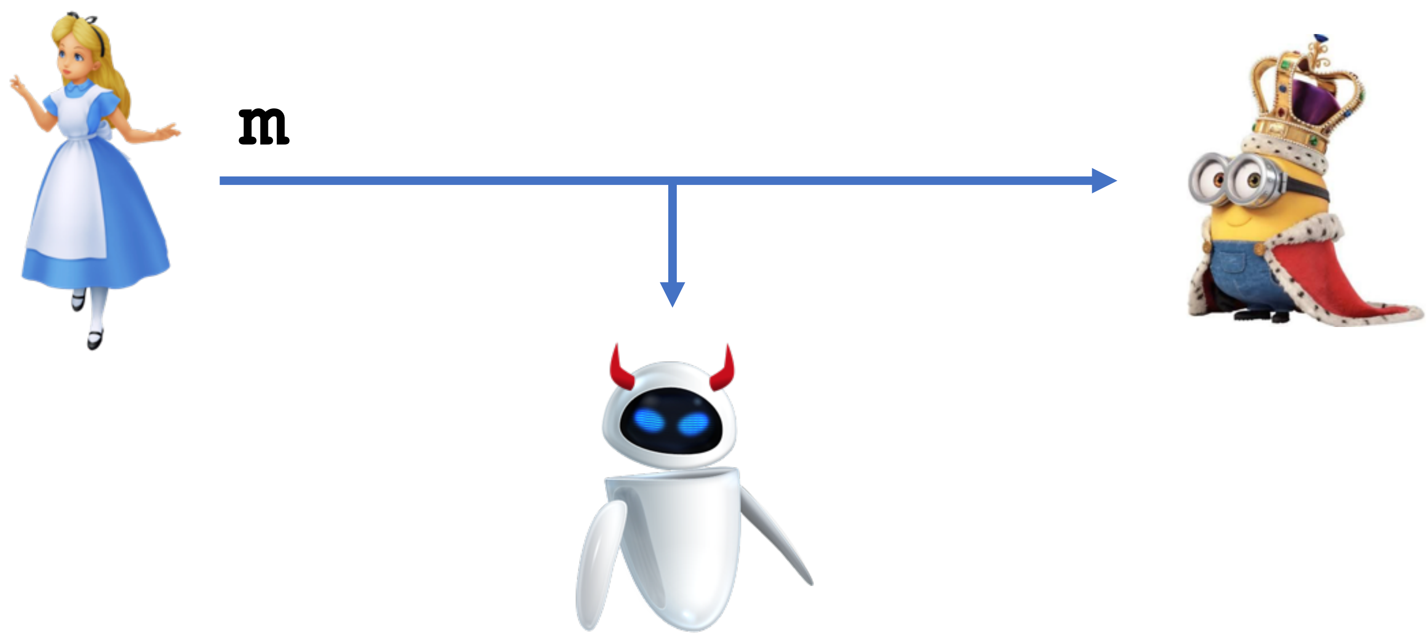
Today

"Pre-modern" Crypto Part I: Substitution Ciphers

Pre-modern Cryptography

1900 B.C. – mid 1900's A.D.

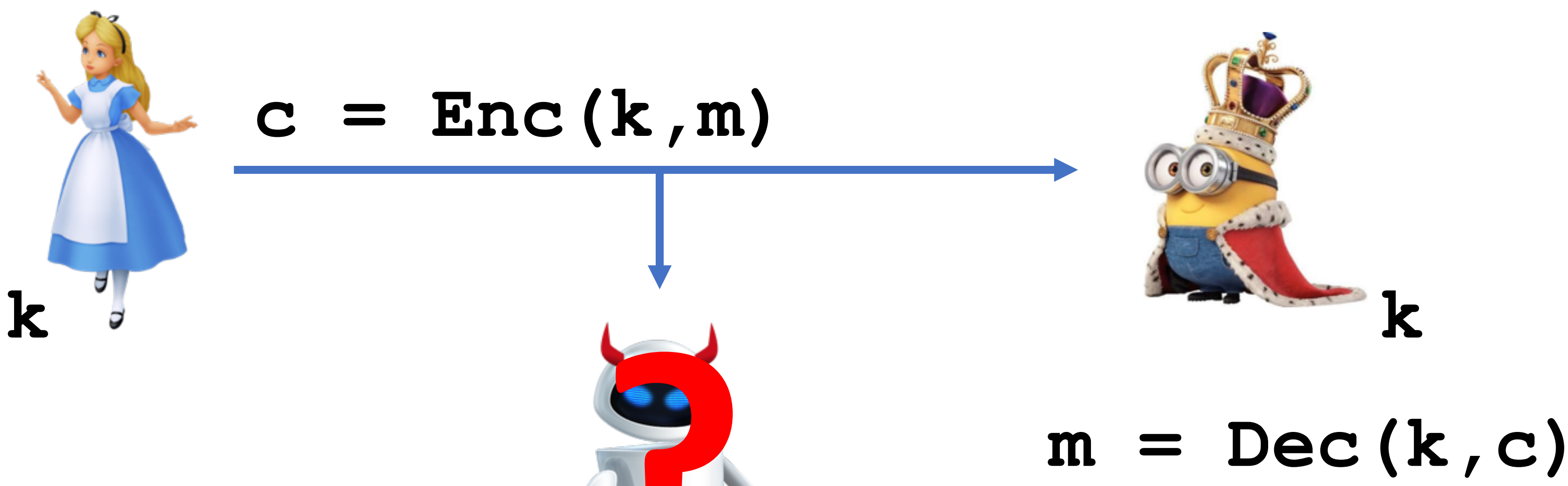
With few exceptions, synonymous with **encryption**



Pre-modern Cryptography

1900 B.C. – mid 1900's A.D

With few exceptions, synonymous with **encryption**



For our discussions, assume **Enc**, **Dec** known, only **k** is secret

Ancient Crypto

1900 BC, Egypt

1500 BC, Mesopotamia

50 B.C. – Caesar Cipher

Used by Julius Caesar

Alphabet shift by 3

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Example:

plaintext: **super secret message**

ciphertext: **VXSHU VHFUHW PHVVDJH**

Caesar not a true cipher: what's the secret key?

Generalization: Shift Ciphers

Shift by fixed, secret increment (**`k = 0,…,25`**)

Some examples:

- Shift by 1: Augustus Caesar; Jewish mezuzah
- Shift by 3: Caesar Cipher
- Shift by 13: ROT13

Sometimes also called “Caesar ciphers”

Security of Shift Ciphers?

Problem: only 26 possibilities for key

“Brute force” attack:

- Try all 26 possible shifts
- For each shift, see if something sensible comes out

Example Brute Force Attack

Ciphertext: **HJETG HTRGTI BTHHPVT**

Key	Plaintext
0	HJETG HTRGTI BTHHPVT
1	IKFUH IUSHUJ CUIIQWU
2	JLGVI JVTIVK DVJJRXV
3	KMHWJ KWUJWL EWKKSYW
4	LNIXK LXVKXM FXLLTZX
5	MOJYL MYWLYN GYMMUAY
6	NPKZM NZXMZO HZNNVBZ
7	OQLAN OAYNAP IAOOWCA
8	PRMBO PBZOBQ JBPPXDB
9	QSNCP QCAPCR KCQQYEC
10	RTODQ RDBQDS LDRRZFD
11	SUPER SECRET MESSAGE
12	TVQFS TFDSFU NFTTBHF

Key	Plaintext
13	UWRGT UGETGV OGUUCIG
14	VXSHU VHFUHW PHVVDJH
15	WYTIV WIGVIX QIWWEKI
16	XZUJW XJHWJY RJXXFLJ
17	YAVKX YKIXKZ SKYYGMK
18	ZBWLY ZLJYLA TLZZHNL
10	ACXMZ AMKZMB UMAAIOM
20	BDYNA BNLANC VNBBJPN
21	CEZOB COMBOD WOCCKQO
22	DFAPC DPNCPE XPDDLRP
23	EGBQD EQODQF YQEEMSQ
24	FHCRE FRPERG ZRFFNTR
25	GIDSF GSQFSH ASGGOUS

Security of Shift Ciphers?

Problem: only 26 possibilities for key

"Brute force" attack:

- Try all 26 possible shifts
- For each shift, see if something sensible comes out

To avoid brute force attacks, need large key space

- On modern hardware, typically need #(keys) $\geq 2^{80}$

(Usually choose at least 2^{128}, 2^{256})

Generalization: Substitution Ciphers

Apply fixed permutation to plaintext letters

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
F	M	S	G	Y	U	J	B	T	P	Z	K	E	W	L	Q	H	V	A	X	R	D	N	C	I	O

Example:

plaintext: **`super secret message`**

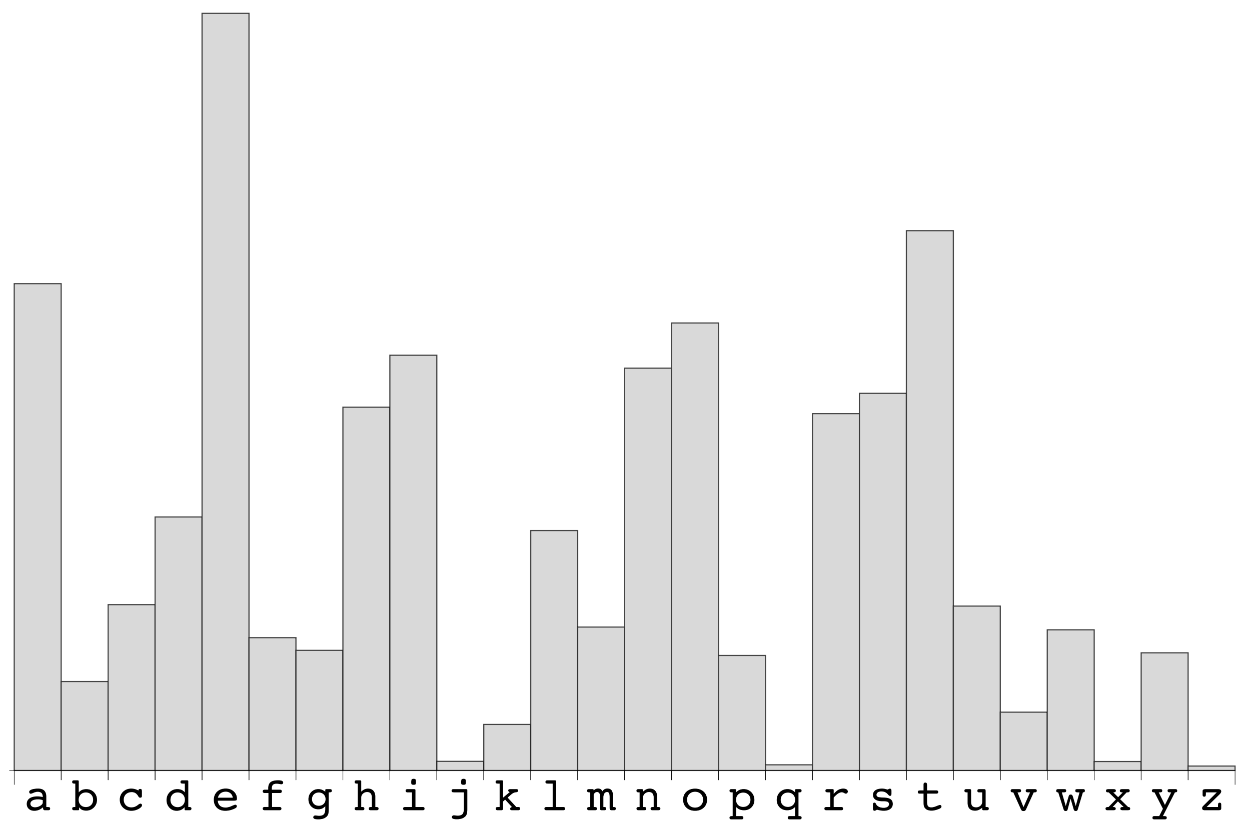
ciphertext: **`ARQYV AYSVYX EYAAFJY`**

Number of possible keys?

$26! \approx 2^{88}$ ➡ brute force attack expensive

800's A.D. – First Cryptanalysis

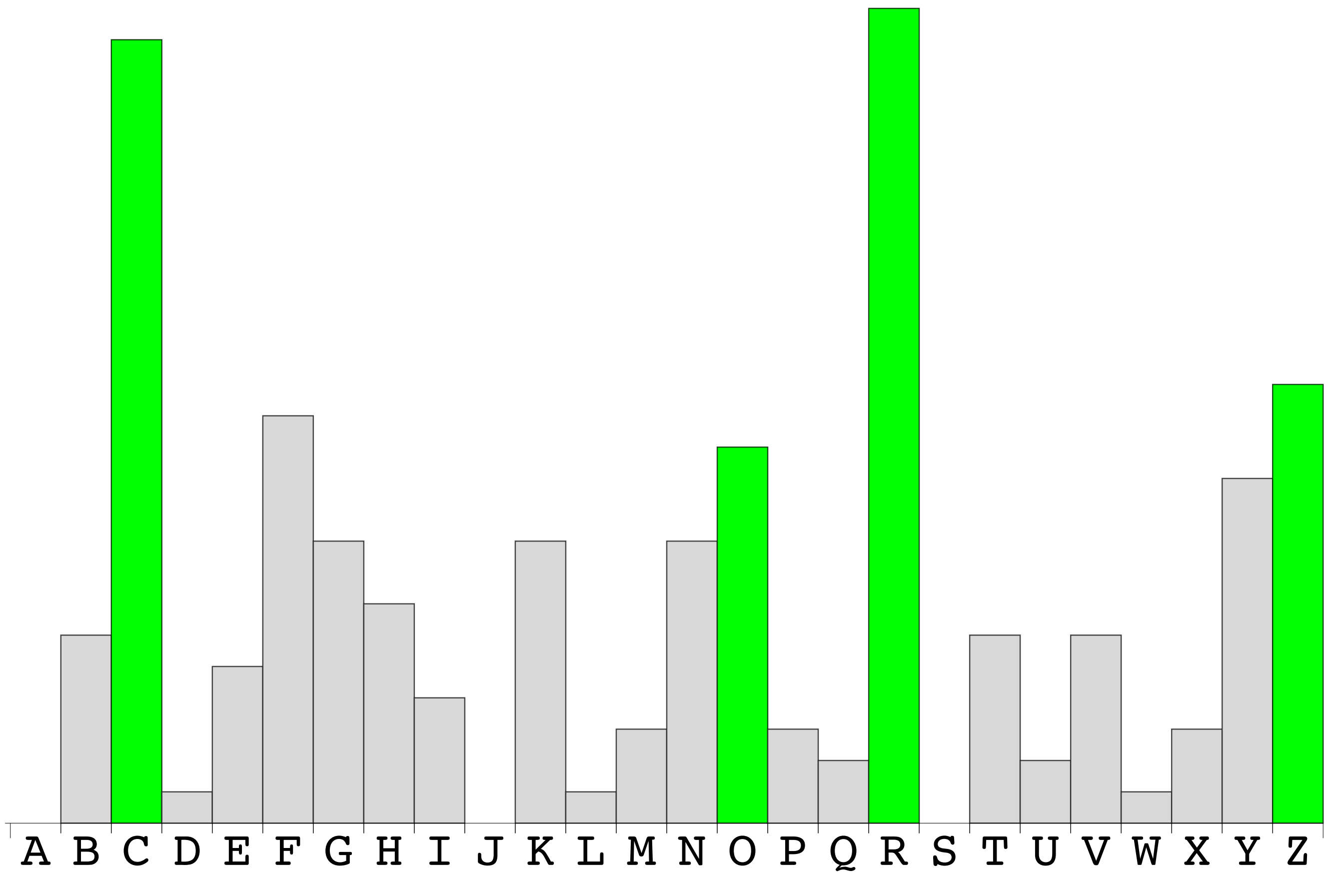
Al-Kindi – Frequency Analysis: some characters are more common than others



Example

BOFC HNR Z NHMNCYCHCYOF KYIVRG CO RFKOBR NRFNYCYPR BZCZ, RPRF CVOHXV CVRE ZGR GRNYTYRFC CO Z MGHCR WOGKR ZCCZKU. YFBRRB, ME KOHFCYFX TRCCRGN ZFB KODIZGYFX CO CEIYKZT CRQC, EOH KZF GRKOPRG CVR ITZYFCRQC ZN LRTT ZN CVR URE

Example



Reasonable conjecture:

e→R, t→C, a→Z, o→O

Example

BoFt HNe a NHMNtYtHtYoF KYIVeG to eFKoBe
NeFNYtYPe Bata, ePeF tVoHXV tVeE aGe
GeNYTYeFt to a MGHte WoGKe attaKU.
YFBeeB, ME KoHFtYFX TetteGN aFB KoDIaGYFX
to tEIYKaT teQt, EoH KaF GeKoPeG tVe
ITaYFteQt aN LeTT aN tVe UeE

Maybe "**data**"?

Maybe "**attack**"?

Probably "**the**"

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Z				R										O					C						

Example

doFt HNe a NHMNtYtHtYoF cYIheG to eFcode NeFNYtYPe data, ePeF thoHXh theE aGe GeNYTYeFt to a MGHte WoGce attack. YFdeed, ME coHFtYFX TetteGN aFd coDIaGYFX to tEIYcaT teQt, EoH caF GecoPeG the ITaYFteQt aN LeTT aN the keE

"encode"?

"and"?

"as"?

"are"?

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Z		K	B	R			V			U				O					C						

Example

“use”?

dont Hse a sHMstYtHtYon cYIher to encode
sensYtYPe data, ePen thoHXh theE are
resYTYent to a MrHte Worce attack.
Yndeed, ME coHntYnX TetteGs and coDIarYnX
to tEIYcaT teQt, EoH can recoPer the
ITaYnteQt as LeTT as the keE

“indeed”? “even”?

“force”? “recover”?

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Z		K	B	R			V			U			F	O			G	N	C						

Example

dont use a suMstitution ciIher to encode sensitive data, even thouXh theE are resiTient to a Mrute force attack. indeed, ME countinX Tetters and coDIarinX to tEIicaT teQt, Eou can recover the ITainteQt as LeTT as the keE

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Z		K	B	R	W		V	Y		U			F	O			G	N	C	H	P				

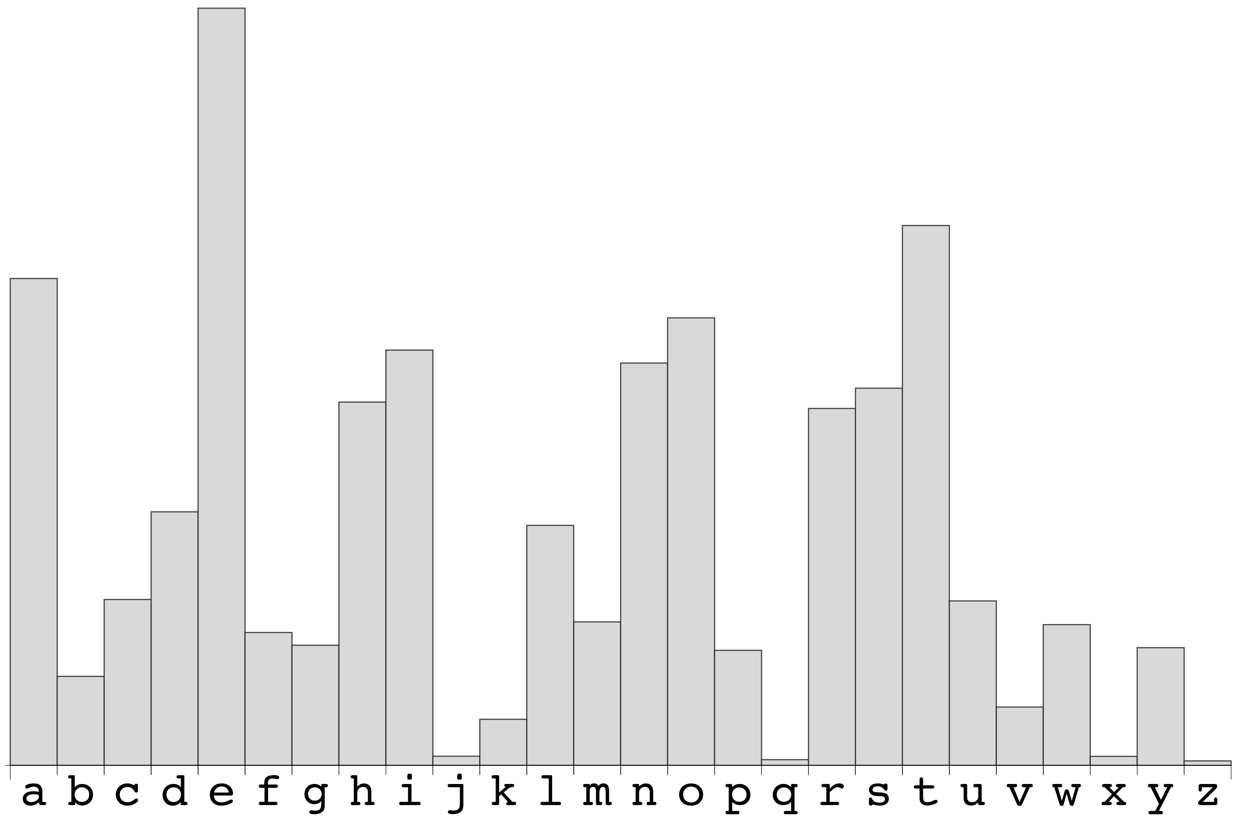
Example

dont use a substitution cipher to encode sensitive data, even though they are resilient to a brute force attack. indeed, by counting letters and comparing to typical text, you can recover the plaintext as well as the key

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Z	M	K	B	R	W	X	V	Y		U	T	D	F	O	I		G	N	C	H	P	L	Q	E	

Problem with Substitution

Differing letter frequencies reveal a lot



Polybius Square

	1	2	3	4	5
1	a	b	c	d	e
2	f	g	h	ij	k
3	l	m	n	o	p
4	q	r	s	t	u
5	v	w	x	y	z

plaintext: `s u p e r  s e c r e t  m e s s a g e`

ciphertext: `4345351542 431513421544 32154343112215`

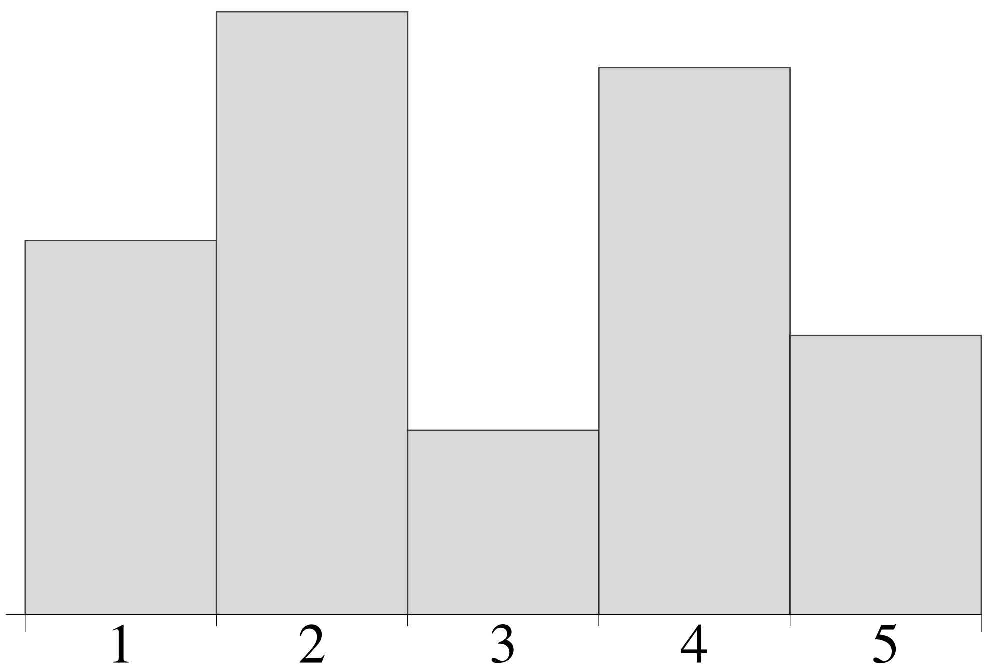
Keyed Polybius Square

	1	2	3	4	5
1	y	n	r	b	f
2	d	l	w	o	g
3	s	p	a	t	k
4	h	v	ij	x	c
5	q	u	z	e	m

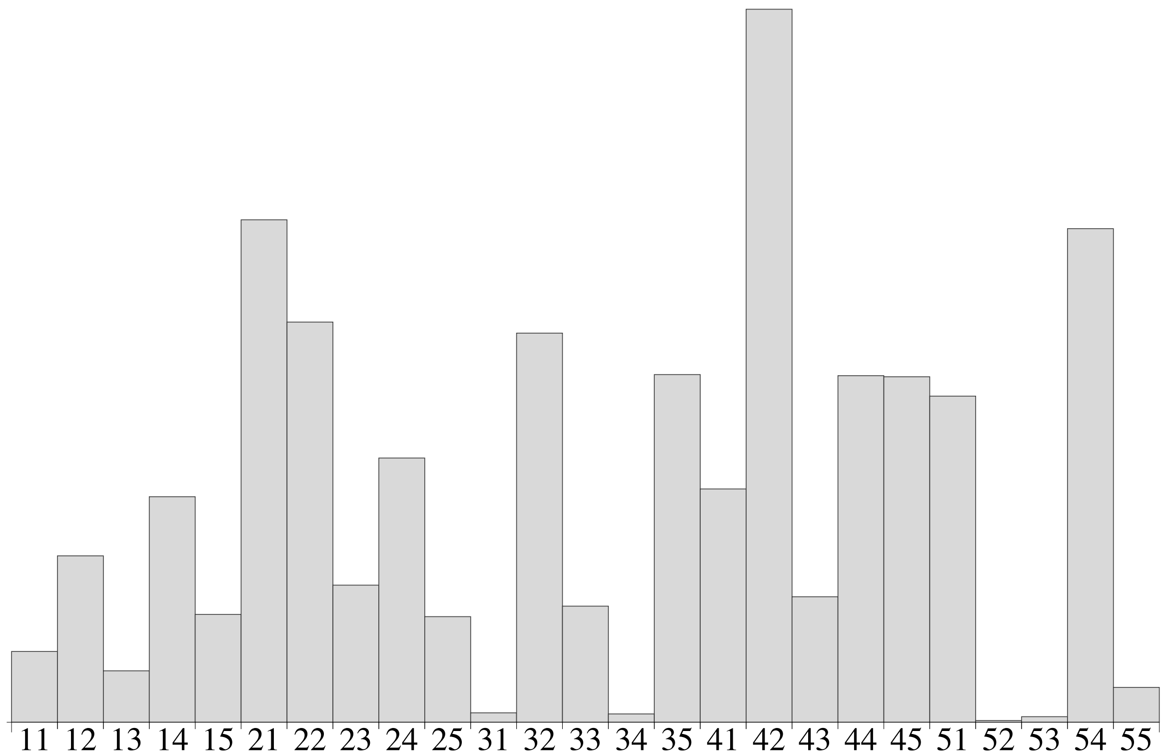
plaintext: `s u p e r  s e c r e t  m e s s a g e`

ciphertext: `3152325413 315445135434 55543131332554`

Frequency of Polybius?



Frequency of Polybius?



General Alphabets

Ptxt and ctxt need not be the same

- ctxt symbols can be letters, (tuples of) numbers, etc.
- ptxt symbols can also numbers, bits, bytes

In general, changing ctxt alphabet doesn't affect security of cipher

- Keyed Polybius = Un-keyed Polybius + Substitution

Other reasons to change ciphertext alphabet?

Polygraphic Substitution

Frequency analysis requires seeing many copies of the same character/group of characters

Idea: encode **d= 2,3,4**, etc characters at a time

- New alphabet size: 26^d
- Symbol frequency decreases:
  - Most common

digram:	"th",	3.9%
trigram:	"the",	3.5%
quadrigram:	"that",	0.8%

- Require much larger ciphertext to perform frequency analysis

Polygraphic Substitution

Example: Playfair cipher

- Invented by Sir Charles Wheatstone in 1854
- Used by British until WWII

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

Polygraphic Substitution

Example: Playfair cipher

- Invented by Sir Charles Wheatstone in 1854
- Used by British until WWII

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

TH

- To encode, choose opposite corners of rectangle

Polygraphic Substitution

Example: Playfair cipher

- Invented by Sir Charles Wheatstone in 1854
- Used by British until WWII

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

TH → XS

- To encode, choose opposite corners of rectangle
- Additional rules for repeats, digrams in same row, etc

Polygraphic Substitution

Limitations:

- For small **d**, frequency analysis still possible given enough ciphertext material

- For large **d**, need $> 26^d$ bits to write down general substitutions
  - Impractical to use arbitrary permutations for large **d**
  - Some tricks (like Playfair) possible to reduce key size while minimizing risk of frequency analysis

Homophonic Substitution

Ciphertexts use a larger alphabet

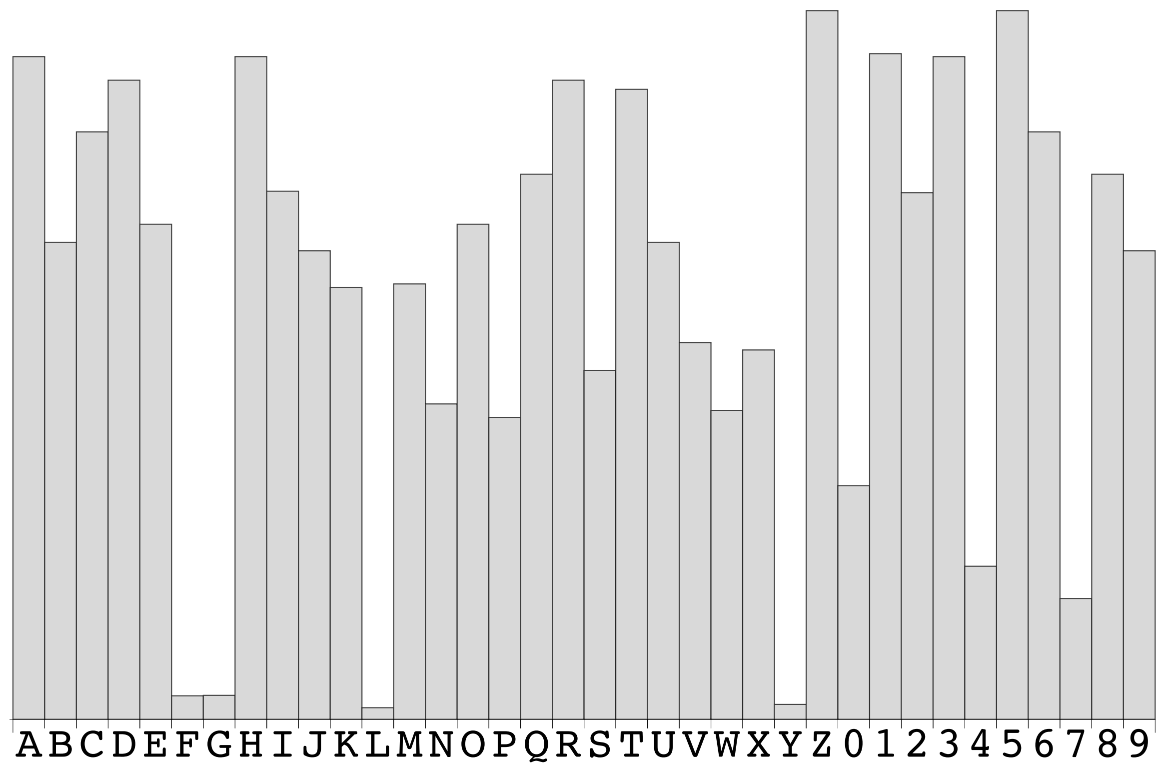
Common letters have multiple encodings

To encrypt, choose encoding at random

plaintext: **`super secret message`**
ciphertext: **`EKPH9 O3MJ3Z VAOEDNH`**

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	0	M	1	A	S	N	U	Q	G	7	T	V	I	6	P	Y	9	E	Z	K	4	X	F	W	L
R				H			B	8					2	C			J	O	5						
				3																					

Homophonic Substitution



a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	0	M	1	A	S	N	U	Q	G	7	T	V	I	6	P	Y	9	E	Z	K	4	X	F	W	L
R				H			B	8					2	C			J	O	5						
				3																					

Homophonic Substitution

In principle, by using sufficiently large ciphertext alphabet, character frequencies can be made ≈uniform

$\implies$ Thwarts vanilla frequency analysis

However, still possible to cryptanalyze

- Frequency analysis on tuples of letters will no longer be uniform

Homophonic Substitution

Example: “Grand Chiffre” (Great Cipher)

N	O	P	Q	R	S	T	V	X	Y	Z	&
811 702	117 238 359 500	219 338	407 595	511 733	355 527	340 618	141 163 284 164	205 436	518 639	820	279 448 615 827

genera, l. ux	35
gens	55
ger . . . 575	95
ges	115
gla	155
gle	215
gli	275
glo, ire	335
gna	375
gne . . . 845	435
gni	485
gno	505
gouvern, er, ment	16
gra, ce	405
grand	525
gre	585
gri	625
gro	665
gua	695
gue	785
guerre	825
gui . de . s	895
Ha	26
he	56
hi	156
ho	216
hu	266
haut	326
habi, t, le, tant	486
heur, e, s	656
hier	796

lieu, x	668
limites	708
livre	728
le Roy de	758
le Prince, de	798
le Duc de	838
le Marquis de	858
le Baron de	898
le Sieur de	49
loin	79
lon	139
lors	189
luy . . . 848	259
Ma . . . 868	298
me . . . 779	339
mi	379
mo	439
mu	489
magasin, s	519
main, s	549
mais . . . 159	579
maitre, s	609
mal, ade, s, je, s,	659
mand, er,	679
maniere, s	719
manque, r	739
marche, s	769
marqu, e, r	799
marecha, l, ux	829
mauvais	859
meilleur	879

Ob	19
obei	59
objet, s	69
obliger, ation	89
observ, er, ation	129
obstacle, s	179
obtenir	220
oc, casion	249
ocup, er	289
of	349
office, ier, s	429
offre, s	449
oient	499
oir	529
oire	559
oit	629
ol	669
om	729
on, s	759
ont,	789
op. pose, ition	819
or	849
ordinaire, s.	899
ordonn, er	20
ordre, s	60
or, s, t	100
os, t	130
ou, r	160
outre	210
ouvr.	240
Pa	270

presque	801
preten, dre, tion	30
pretexte	841
pri	881
principa, l, ux	52
prisonnier, s	132
pro	162
prochain	202
profit, er	262
projet, s	282
propos, ition, s	382
provision, s	422
prouv	442
pru	462
publi, er, c	512
puis, sance	572
Qu	612
qua	672
qualite	722
quand	742
quantite	762
quarente	782
quart, ier, s	822
quatre	842
que	862
quel, le, s	882
question, s	23
qui . . . 50	53
qu'il	73
quinze	133
quo, n . . . 390	153

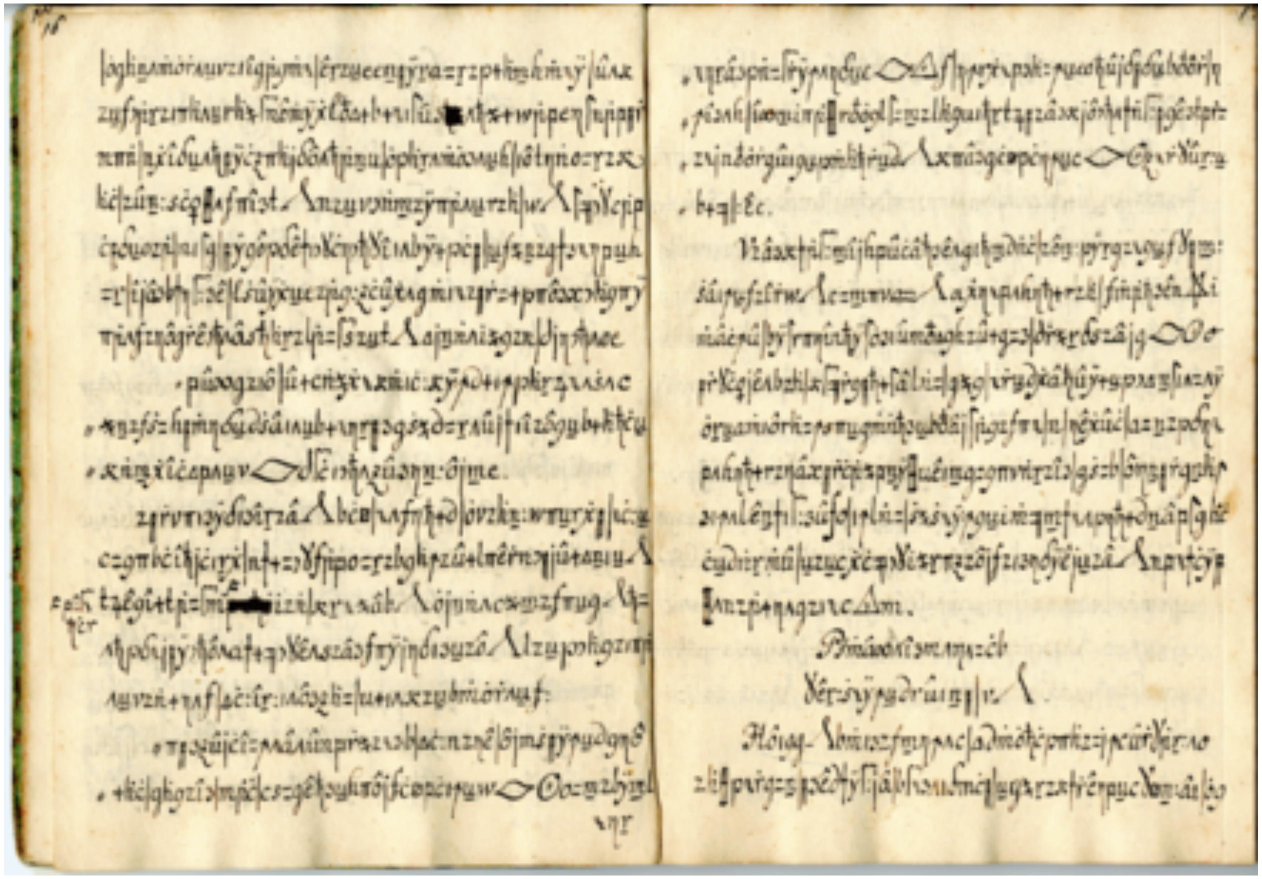
Homophonic Substitution

Example: “Grand Chiffre” (Great Cipher)

- Developed in 1600’s, used by Louis XIV
- Remained unbroken for 200 years
- Combination of polygraphic and homophonic
- 1890’s - finally cracked by Étienne Bazeries
    - Guessed that “`124-22-125-46-345`” stood for “`les ennemies`”
    - From there, things unraveled

Homophonic Substitution

Example: Copiale cipher



Homophonic Substitution

Example: Copiale cipher

- 105-page encrypted book written in 1730’s
- Secret society of German ophthalmologists
- Not broken until 2011 with help of computers

Polyalphabetic Substitution

Use a different substitution for each position

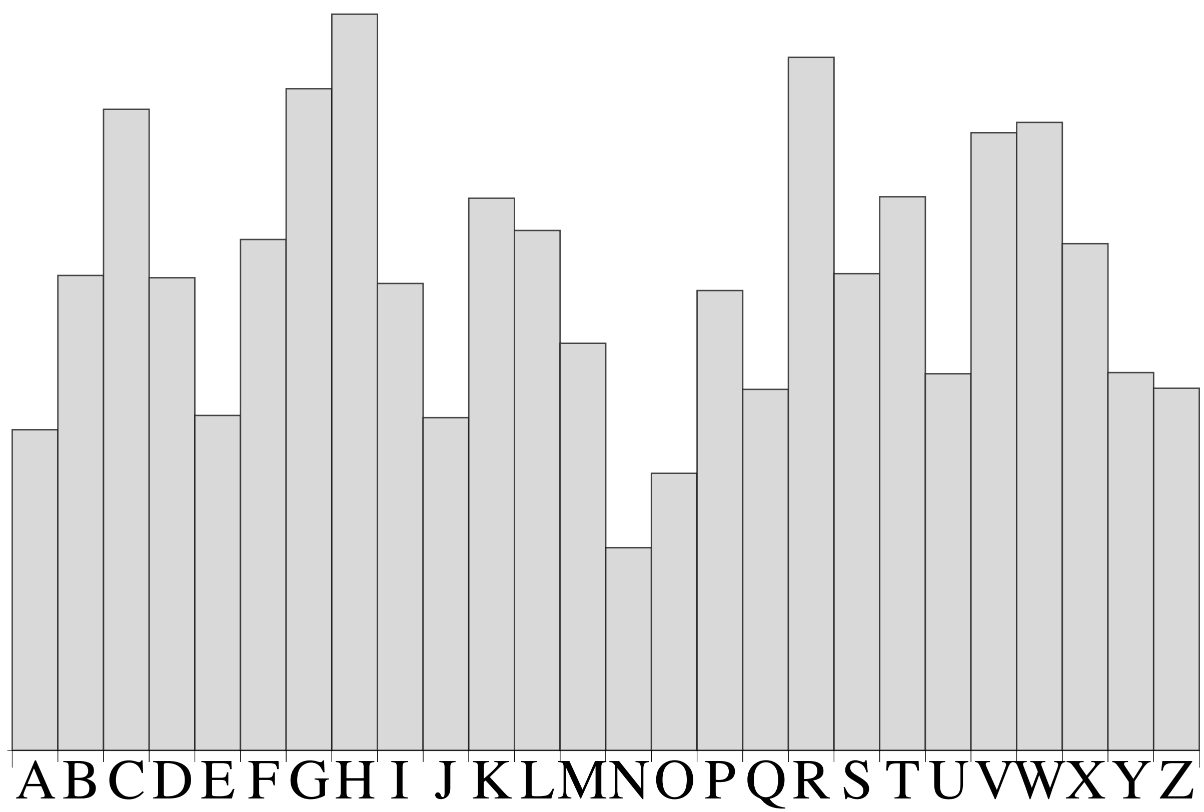
Example: Vigenère cipher

- Sequence of shift ciphers defined by keyword

keyword:	`crypt ocrypt ocrypto`
plaintext:	`super secret message`
ciphertext:	`ULNTK GGTPTM AGJQPZS`

Polyalphabetic Substitution

Vanilla frequency analysis gives average of several substitution ciphers



Cryptanalysis of Vigenère

Suppose we know keyword length

- Group letters into **n** buckets, each bucket encrypted using the same shift
- Perform frequency analysis on each bucket

Suppose we don't know keyword length

- Brute force: try several lengths until we get the right one
- Improvement: Kasiski examination, superposition

Kasiski Examination

Published 1863, apparently known to Babbage as early as 1840's

Example:

key: `cryptocryptocryptocryptocryptocryptocrypto`

ptxt: `acannercancanasmanycansasacannercancancans`

ctxt: `CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG`

All **RED**/**PURPLE** chunks are multiples of **6** apart

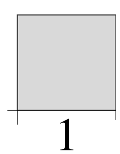
- Good indication that the key length is **1,2,3**, or **6**

Superposition

Compare shifts of ciphertext, looking for shifts containing many matches

Example: shift by 1

```
CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG
 CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG
```

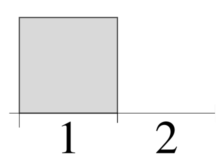


Superposition

Compare shifts of ciphertext, looking for shifts containing many matches

Example: shift by 2

```
CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG
  CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG
```

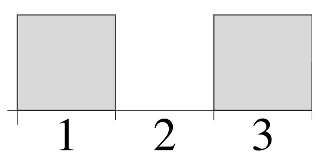


Superposition

Compare shifts of ciphertext, looking for shifts containing many matches

Example:shift by 3

```
CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG
   CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG
```

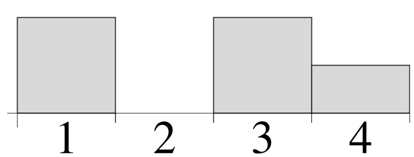


Superposition

Compare shifts of ciphertext, looking for shifts containing many matches

Example: shift by 4

```
CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG
    CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG
```



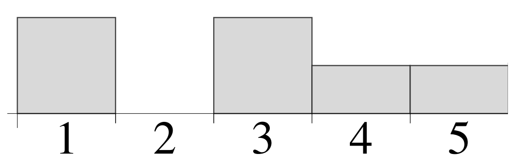
Superposition

Compare shifts of ciphertext, looking for shifts containing many matches

Example: shift by 5

CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG

CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG



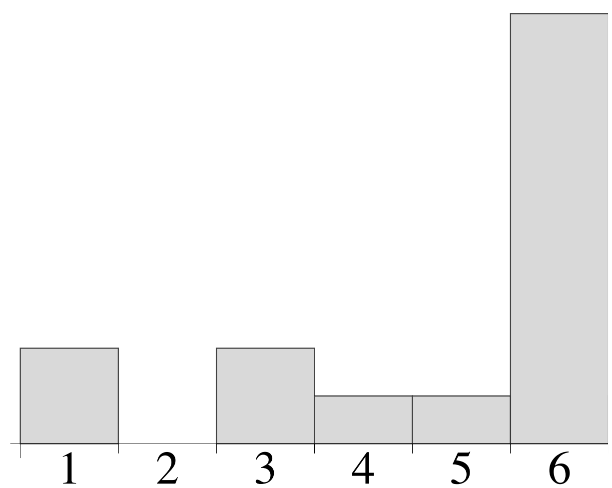
Superposition

Compare shifts of ciphertext, looking for shifts containing many matches

Example: shift by 6

CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG

        CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG



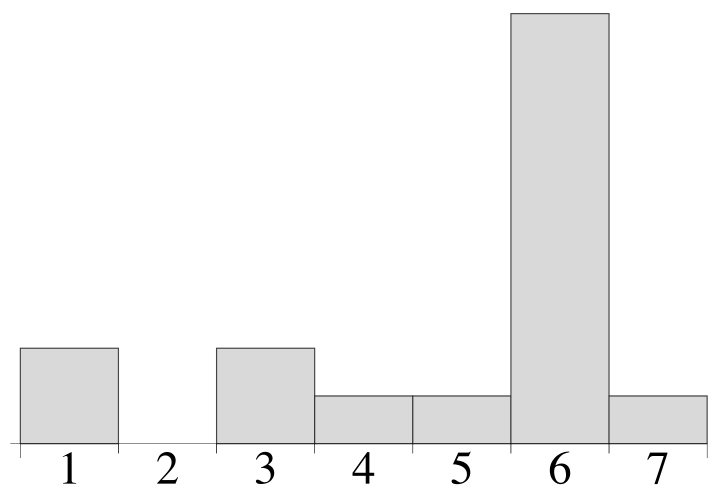
Superposition

Compare shifts of ciphertext, looking for shifts containing many matches

Example: shift by 7

CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG

       CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG



Superposition

Why does it work?

For shifts that are multiples of key size:

- Both bottom and top ciphertexts encrypted with same key
- **#(ctxt matches) = #(ptxt matches)**
  ≈ |ptxt| * col. prob. for English
  ≈ |ptxt| * 0.065

Superposition

Why does it work?

For shifts that are NOT multiplies of key size:

- Both bottom and top ciphertexts encrypted with "independent" shifts
- Probability of a match at any position is **1/26 ≈ 0.038**
- **#(ctxt matches) ≈ |ptxt| * 0.038**

The One-Time Pad

Vigenère on steroids

- Every character gets independent substitution
- Only use key to encrypt one message,
  key length ≥ message length

keyword:	**`agule melpqw gnspemr`**
plaintext:	**`super secret message`**
ciphertext:	**`SAIPV EINGUP SRKHESR`**

No substitution used more than once, so frequency analysis is impossible

The One-Time Pad

1882: described by Frank Miller for the telegraph

- Words and phrases first converted to 5-digit numbers using a codebook
- Key = sequence of “shift-numbers” to be added to resulting digits

1919: Patent for Vernam cipher

- Map characters to 5-bit strings using Baudot code
- Bitwise XOR with key = random bit string

Advantages

1945: Claude Shannon proved that if:

- A truly random key is used
- The key is only used to encrypt only one message
- And the key is longer than the message

Then the scheme is *perfectly secure*

Notation

Two random variables $\mathbf{X,Y}$ over a finite set $\mathbf{S}$ have identical distributions if, for all $\mathbf{s} \in \mathbf{S}$,

$$\Pr[\ X = s] \;=\; \Pr[\ Y = s]$$

In this case, we write

$$X \stackrel{d}{=} Y$$

Perfect Secrecy [Shannon'49]

Definition: A scheme **(Enc,Dec)** has **perfect secrecy** if, for any two messages $\mathbf{m_0}, \mathbf{m_1} \in \mathbf{M}$

$$\mathbf{Enc(K,\ m_0)} \stackrel{d}{=} \underbrace{\mathbf{Enc(K,\ m_1)}}$$

Random variable corresponding to uniform distribution over **K**

Random variable corresponding to encrypting $\mathbf{m_1}$ using a uniformly random key

Perfect Secrecy of One-time Pad

For concreteness, use XOR (Vernam cipher); applies equally well to other variants of one-time pad

Key space $\mathbf{K} = \{0,1\}^n$
Message space $\mathbf{M} = \{0,1\}^n$
Ciphertext space $\mathbf{C} = \{0,1\}^n$

$\mathbf{Enc(k, m) = k \oplus m}$
$\mathbf{Dec(k, c) = k \oplus c}$

Perfect Secrecy of One-time Pad

Theorem: For any message $\mathbf{m} \in \{0,1\}^n$ and ciphertext $\mathbf{c} \in \{0,1\}^n$,

$$\Pr[\, \mathrm{Enc}(k, m) = c] = 2^{-n}$$

Proof:

$$\begin{aligned} \Pr[\, \mathrm{Enc}(k, m) = c] \quad &= \Pr[\, k \oplus m = c \,] \\ &= \Pr[\, k = c \oplus m \,] \\ &= 2^{-n} \end{aligned}$$

Limitations of One-time Pad

Need extremely large random keys and secure way to transmit them!

5-UCO British OTP system (WWII)

- Key tape for single unit cost £5,000 a year (~$300k in 2018 dollars)

German GEE (WWII)

- Key’s not truly random, cryptanalyzed by US Army

Russian diplomatic OTP (WWII, Cold Ward)

- Tapes occasionally re-used, successful cryptanalysis by US and UK intelligence

Cryptanalysis of OTP

Try to encrypt two messages, security will fail

$$\begin{aligned} \mathrm{Enc}(k,m_0) \oplus \mathrm{Enc}(k,m_1) \\ &= (k \oplus m_0) \oplus (k \oplus m_1) \\ &= m_0 \oplus m_1 \end{aligned}$$

Enough redundancy in English text to usually recover messages from XOR

Project 1: Cryptanalysis

Project 1: Cryptanalysis

Setup: you're an intern at a super secret intelligence agency, which is trying to decrypt a batch of documents

What you know:
- All pencil-and-paper ciphers
- All based on schemes we'll see this week

Your task:
- Use what you've learned to decrypt the documents

Part 0: Form Teams

Due: Friday February 9th

Instructions:

- Teams of 2-3 people
- Sign up on Google spreadsheet
- Use Piazza team-finding feature if necessary

Documents will be released to teams by **Feb 10th**

Part 1: Basic Analysis (15%)

Due: Tuesday February 20th

Instructions: tell us as much as possible about each document

- Which documents encrypted by same means?
- What cipher used?
- Parameters of cipher (key length, etc)

Main purpose is to give early feedback

Part 2: Cryptanalysis (85%)

Due: Tuesday March 6th

Instructions: Actually decrypt the documents

- Also, give thorough write-up on your methodology
- Also, report on intelligence gathered

For both parts 1 and 2, you should definitely make use of computers to perform analysis

- Please submit your code

Competition

Submit any (partially) decrypted documents early and often

Every Monday morning, teaching staff will test how well you've done so far

- Most successful team will receiver **2 bonus points**
- Runner up will receive **1 bonus point**

Bonus dates:

Feb 19th, Feb 26th, March 5th

Reminders

By Friday Feb 9th:

- HW0: Fill out OH Doodle poll
- Find Teams for Project 1

Due Tuesday Feb 13th:

- HW1, on course webpage