Private Key Encryption

Presentation by:
Dr. V. Balasubramanian
SSN College of Engineering

Objectives

- Focus on formal study of Modern Cryptography
- Understand real-world crypto via a rigorous approach

Classical Cryptography

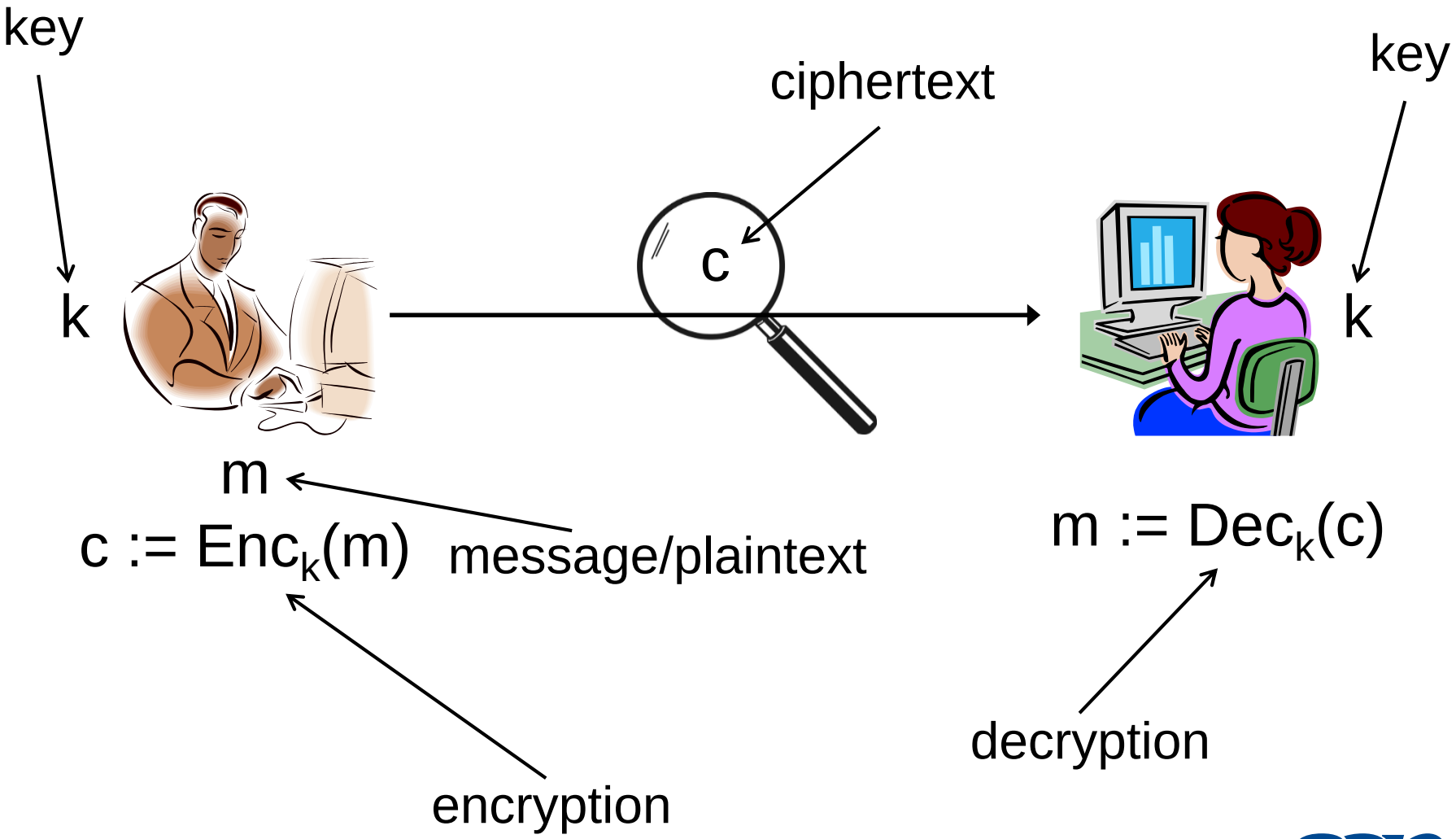
"…the art of writing or solving codes…"

- Historically, cryptography focused exclusively on ensuring *private communication* between two parties sharing secret information in advance (using "codes" aka *private-key encryption*)

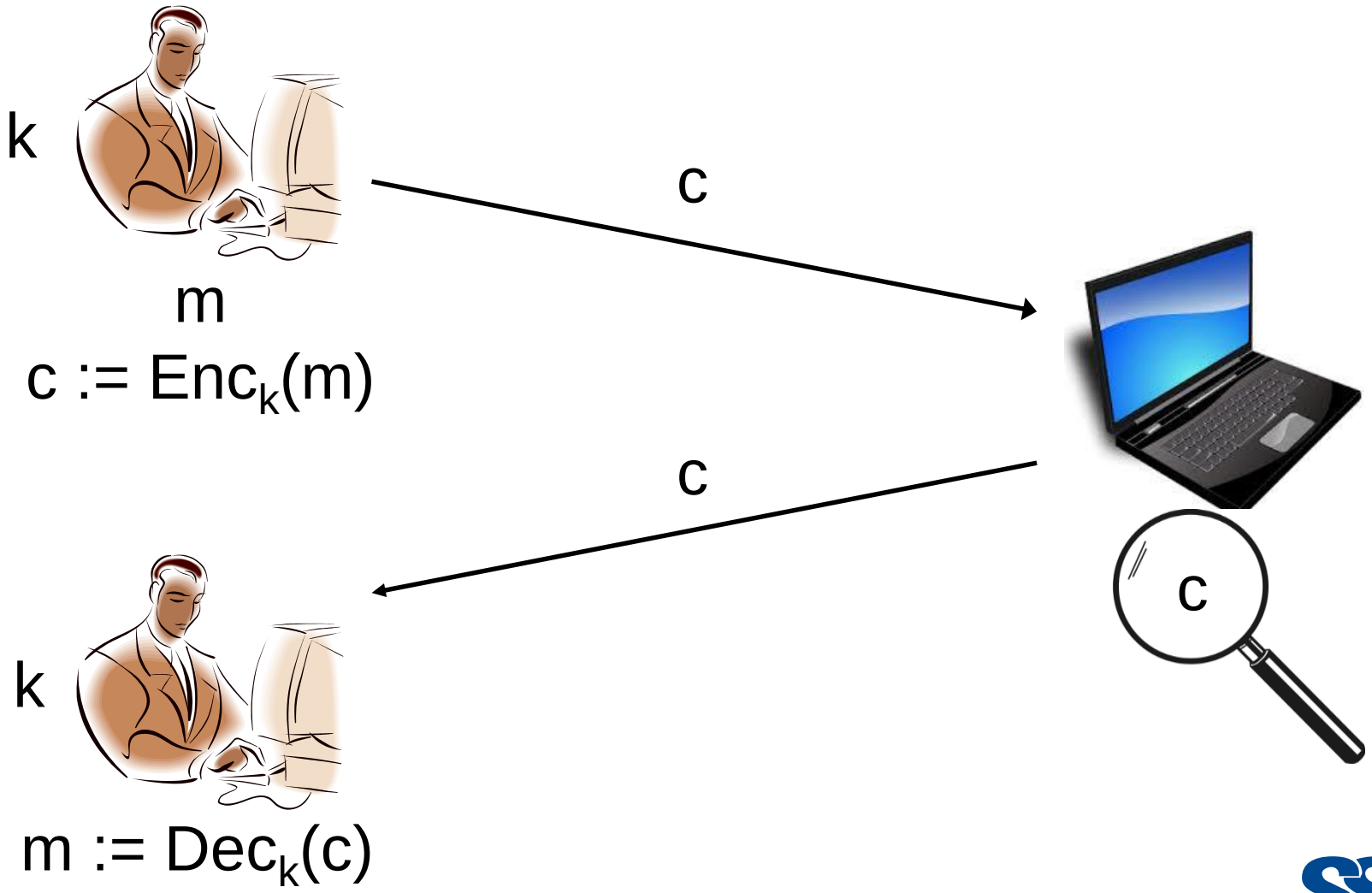
Cryptography

- Security of all classical encryption schemes relied on a secret—a key—shared by the communicating parties in advance and unknown to the eavesdropper.
- This scenario is known as the private-key (or shared-/secret-key) setting, and private-key encryption is just one example of a cryptographic primitive used in this setting.

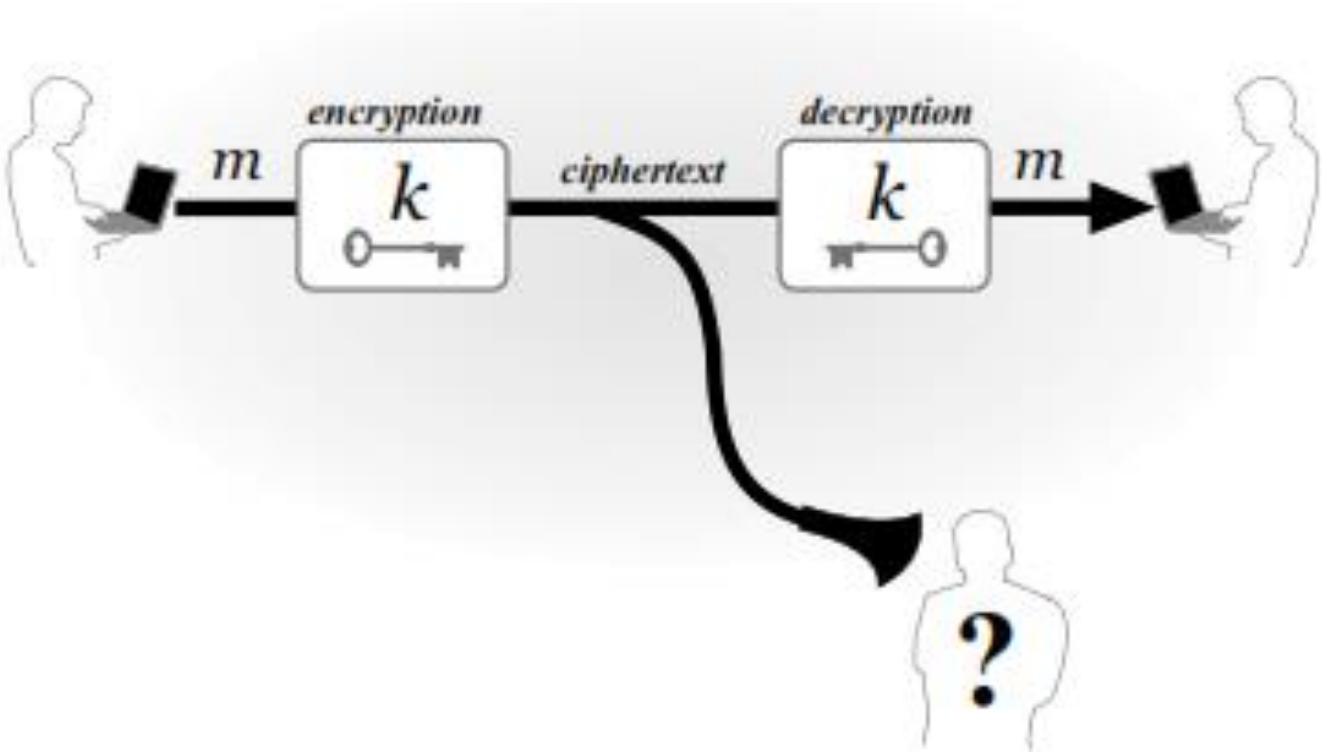
Private-key encryption



Private-key encryption

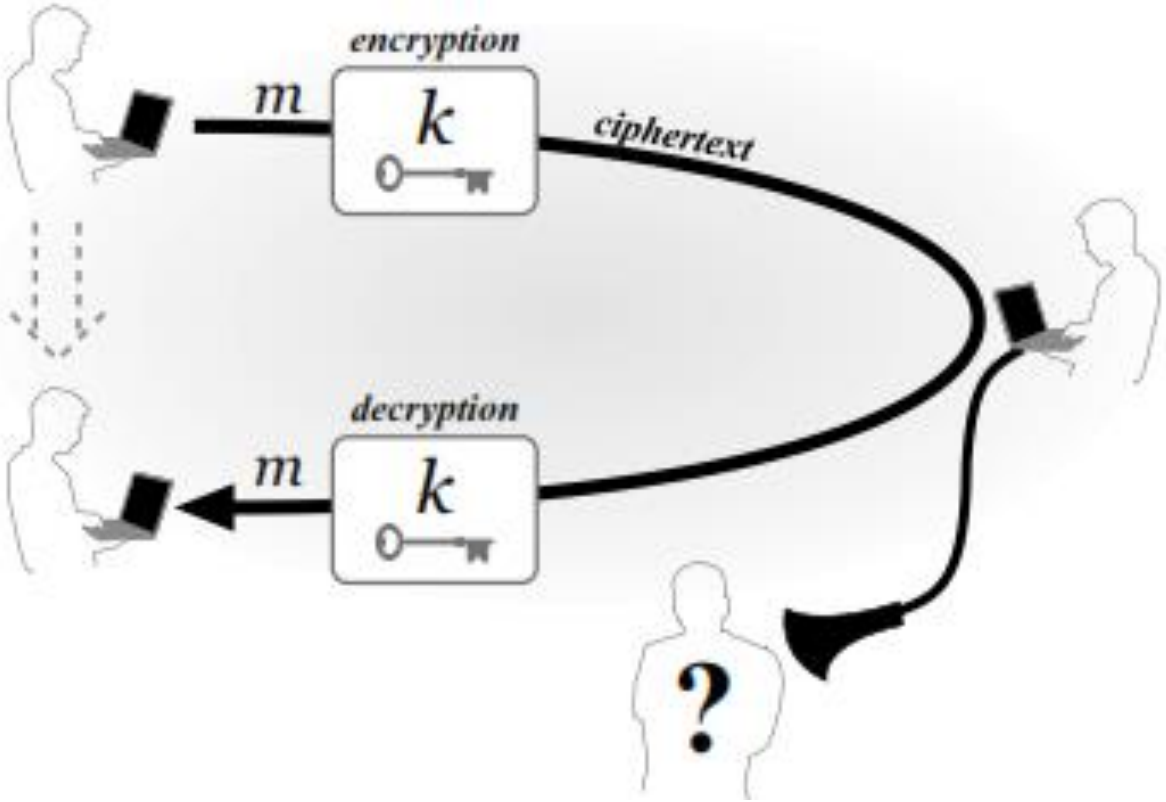


Applications

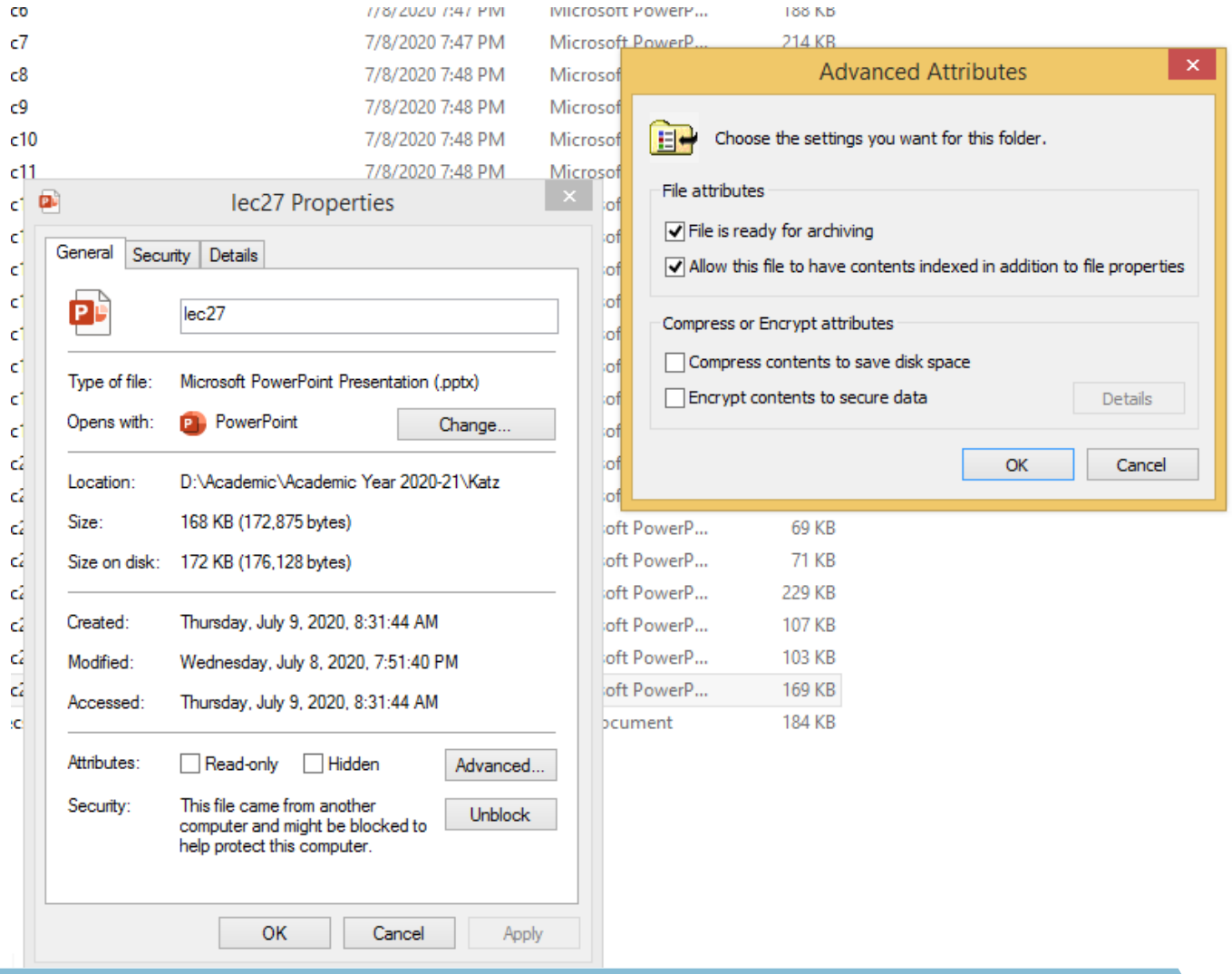


Two Party Communication

Application



Application Disk Encrypt



Private-key encryption

- A *private-key encryption scheme* is defined by a message space $\mathcal{M}$ and algorithms (Gen, Enc, Dec):
  - Gen (key-generation algorithm): outputs $k \in \mathcal{K}$
  - Enc (encryption algorithm): takes key k and message $m \in \mathcal{M}$ as input; outputs ciphertext c
    $$c \leftarrow \text{Enc}_k(m)$$
  - Dec (decryption algorithm): takes key k and ciphertext c as input; outputs m or "error"

For all $m \in \mathcal{M}$ and k output by Gen,

$$\text{Dec}_k(\text{Enc}_k(m)) = m$$

Algorithms

1. The *key-generation algorithm* Gen is a probabilistic algorithm that outputs a key k chosen according to some distribution.

2. The *encryption algorithm* Enc takes as input a key k and a message m and outputs a ciphertext c. We denote by $\mathsf{Enc}_k(m)$ the encryption of the plaintext m using the key k.

3. The *decryption algorithm* Dec takes as input a key k and a ciphertext c and outputs a plaintext m. We denote the decryption of the ciphertext c using the key k by $\mathsf{Dec}_k(c)$.

Kerckhoffs's principle

- *The encryption scheme* is not secret
  - The attacker knows the encryption scheme
  - The only secret is the *key*
  - The key must be chosen at random; kept secret

- Some arguments in favor of this principle
  - Easier to keep *key* secret than *algorithm*
  - Easier to change *key* than to change *algorithm*
  - Standardization
    - Ease of deployment
    - Public validation

The shift cipher

- Consider encrypting English text
- Associate ‘a’ with 0; ‘b’ with 1; …; ‘z’ with 25
- $k \in \mathcal{K} = \{0, \ldots, 25\}$

```
helloworldz
ccccccccccc
jgnnqyqtnfb
```

- To encrypt using key k, shift every letter of the plaintext by k positions (with wraparound)
- Decryption just does the reverse

Caesar’s Cipher

- Caesar’s cipher. One of the oldest recorded ciphers, known as Caesar’s cipher.
- a was replaced with D, b with E, and so on
- Message = begintheattacknow
- Cipher = EHJLQWKHDWWDFNQRZ

Modular arithmetic

- x = y mod N if and only if N divides x-y
- [x mod N] = the remainder when x is divided by N
  - I.e., the unique value $y \in \{0, \ldots, N-1\}$ such that
    x = y mod N

- 25 = 35 mod 10
- 25 ≠ [35 mod 10]
- 5 = [35 mod 10]

The shift cipher, formally

- $\mathcal{M}$ = {strings over lowercase English alphabet}
- Gen: choose uniform $k \in \{0, \ldots, 25\}$
- $Enc_k(m_1 \ldots m_t)$: output $c_1 \ldots c_t$, where
  $c_i := [m_i + k \bmod 26]$
- $Dec_k(c_1 \ldots c_t)$: output $m_1 \ldots m_t$, where
  $m_i := [c_i - k \bmod 26]$

- Can verify that correctness holds…

Is the shift cipher secure?

- No -- only 26 possible keys!
  - Given a ciphertext, try decrypting with every possible key
  - Only one possibility will "make sense"
  - (What assumptions are we making here?)
- Example of a "brute-force" or "exhaustive-search" attack

Is Cipher secure

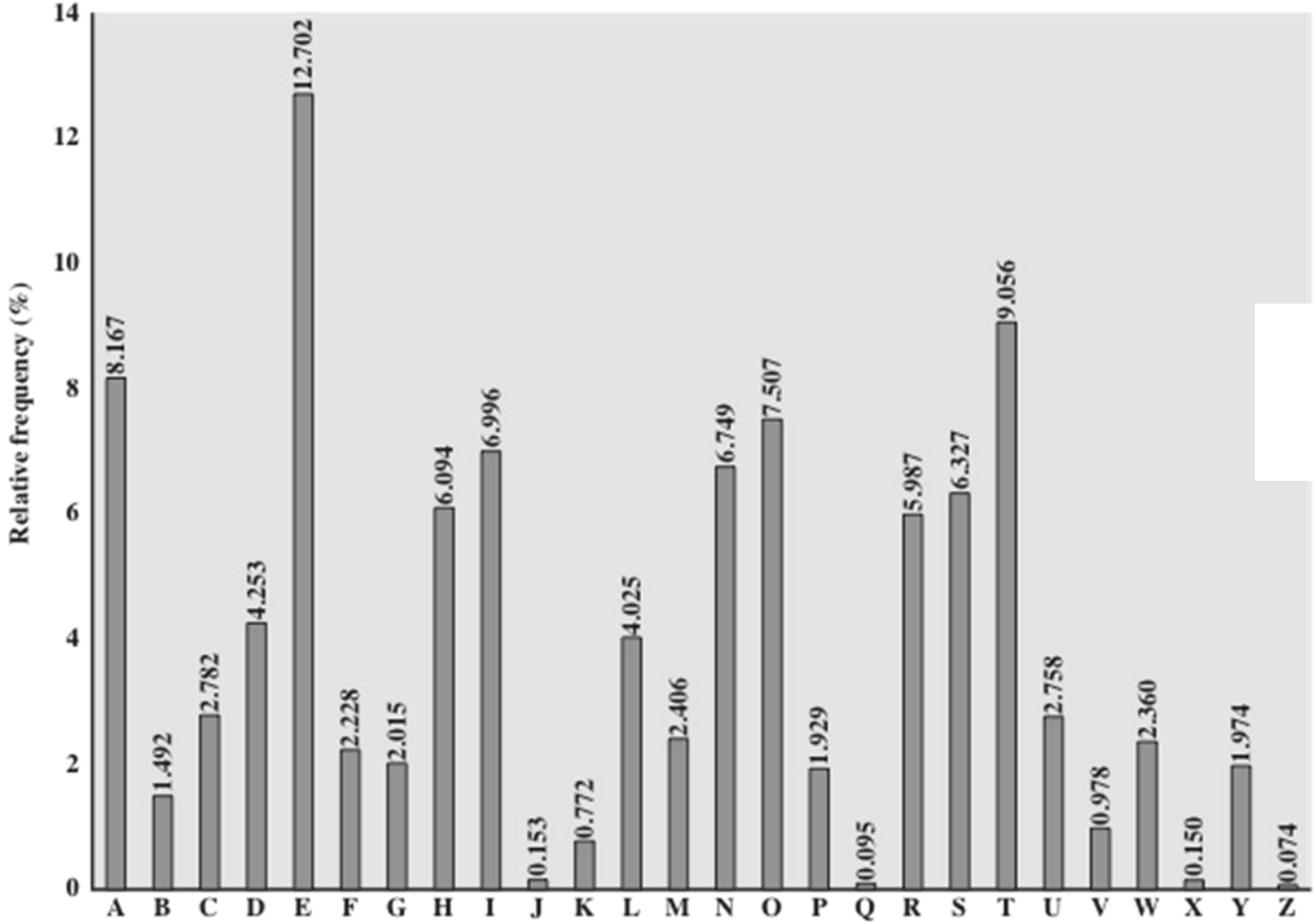
- Nowadays, attackers can use supercomputers, tens of thousands of personal computers, or graphics processing units (GPUs) to speed up brute-force attacks.

Example Brute Force

- Ciphertext `uryybjbeyq`
- Try every possible key…
  - `tqxxaiadxp`
  - `spwwzhzcwo`
  - …
  - `helloworld`

KEY	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

Figure 2.3 Brute-Force Cryptanalysis of Caesar Cipher



$$\sum_{i=0}^{25} p_i^2 \approx 0.065.$$

Figure 2.5 Relative Frequency of Letters in English Text

Byte-wise shift cipher

- Work with an alphabet of *bytes* rather than (English, lowercase) *letters*
  - Works natively for arbitrary data!

- Use XOR instead of modular addition
  - Essential properties still hold

Hexadecimal (base 16)

Hex	Bits ("nibble")	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7

Hex	Bits ("nibble")	Decimal
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Hexadecimal (base 16)

- 0x10
  - 0x10 = 16*1 + 0 = 16
  - 0x10 = 0001 0000
- 0xAF
  - 0xAF = 16*A + F = 16*10 + 15 = 175
  - 0xAF = 1010 1111

ASCII

- Characters (often) represented in ASCII
  - 1 byte/char = 2 hex digits/char

Hex	Dec	Char		Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL	null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH	Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX	Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX	End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT	End of transmission	0x24	36	$	0x44	68	D	0x64	100	d
0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB	Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	[	0x7B	123	{
0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	\	0x7C	124	|
0x1D	29	GS	Group separator	0x3D	61	=	0x5D	93	]	0x7D	125	}
0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

Source: http://benborowiec.com/2011/07/23/better-ascii-table/

ASCII

- '1' = 0x31 = 0011 0001
- 'F' = 0x46 = 0100 0110

- Note that writing 0x00 to a file is different from writing "0x00" to a file
  - 0x00 = 0000 0000 (1 byte)
  - "0x00" = 0x30 78 30 30
    = 0011 0000 0111 1000… (4 bytes)

Useful observations

- Only 128 valid ASCII chars (128 bytes invalid)
- 0x20-0x7E printable
- 0x41-0x7a includes upper/lowercase letters
  - Uppercase letters begin with 0x4 or 0x5
  - Lowercase letters begin with 0x6 or 0x7

Byte-wise shift cipher

- $\mathcal{M}$ = {strings of bytes}
- Gen: choose uniform byte $k \in \mathcal{K} = \{0, \ldots, 255\}$
- $Enc_k(m_1 \ldots m_t)$: output $c_1 \ldots c_t$, where
  $$c_i := m_i \oplus k$$
- $Dec_k(c_1 \ldots c_t)$: output $m_1 \ldots m_t$, where
  $$m_i := c_i \oplus k$$

- Verify that correctness holds…

Code for byte-wise shift cipher

```
// read key from key.txt (hex) and message from ptext.txt (ASCII);
// output ciphertext to ctext.txt (hex)
#include <stdio.h>

main(){
  FILE *keyfile, *pfile, *cfile;
  int i;
  unsigned char key;
  char ch;

  keyfile = fopen("key.txt", "r"), pfile = fopen("ptext.txt", "r"), cfile = fopen("ctext.txt", "w");

  if (fscanf(keyfile, "%2hhX", &key)==EOF) printf("Error reading key.\n");

  for (i=0; ; i++){
    if (fscanf(pfile, "%c", &ch)==EOF) break;
    fprintf(cfile, "%02X", ch^key);
  }

  fclose(keyfile), fclose(pfile), fclose(cfile);
}
```

Is this cipher secure?

- No -- only 256 possible keys!
  - Given a ciphertext, try decrypting with every possible key
  - If ciphertext is long enough, only one plaintext will “make sense”
- Can further optimize
  - First nibble of plaintext likely 0x4, 0x5, 0x6, 0x7 (assuming letters only)
  - Can reduce exhaustive search to 26 keys (how?)

Sufficient key space principle

- The key space must be large enough to make exhaustive-search attacks impractical
  - How large do you think that is?
- Note: this makes some assumptions…
  - English-language plaintext
  - Ciphertext sufficiently long so only one valid plaintext

The Vigenère cipher

- The key is now a *string*, not just a character
- To encrypt, shift each character in the plaintext by the amount dictated by the next character of the key
  - Wrap around in the key as needed
- Decryption just reverses the process

```
tellhimaboutme
cafecafecafeca
veqpjiredozxoe
```

The Vigenère cipher

- Size of key space?
  - If keys are 14-character strings over the English alphabet, then key space has size $26^{14} \approx 2^{66}$
  - If variable length keys, even more…
  - Brute-force search infeasible
- Is the Vigenère cipher secure?
- (Believed secure for many years…)

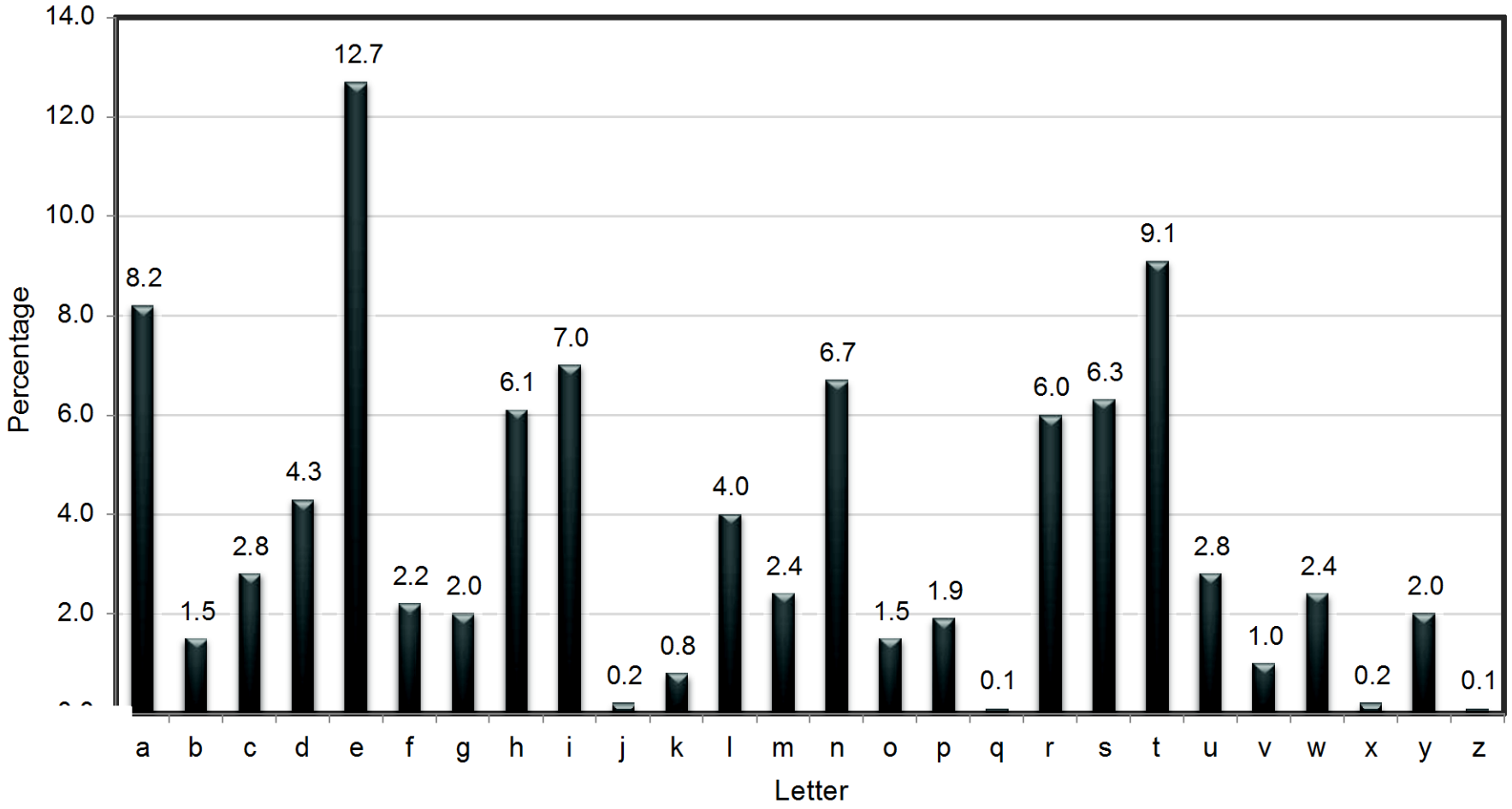
Attacking the Vigenère cipher

- (Assume a 14-character key)
- Observation: every 14th character is "encrypted" using the same shift

```
      veqpjiredozxoeualpcmsdjqu
```

- Look `iqndnossoscdcusoakjqmxpqr` is (alm `hyycjqoqqodhjcciowieii` ext encrypted with the shift cipher
  - Though a direct brute-force attack doesn't work…
  - Why not?

Using plaintext letter frequencies



Attacking the Vigenère cipher

- Look at every 14th character of the ciphertext, starting with the first
  - Call this a "stream"
- Let α be the most common character appearing in this stream
- Most likely, this character corresponds to the most common plaintext character ('e')
  - Guess that the first character of the key is α - 'e'
- Repeat for all other positions

- This is somewhat haphazard…

A better attack

- Let p_i ($0 \le i \le 25$) denote the frequency of the i^{th} English letter in general text
  - One can compute that $\Sigma_i\ p_i^2 \approx 0.065$
- Let q_i denote the observed frequency of the i^{th} letter in a given stream of the ciphertext
- If the shift for a stream is j, expect $q_{i+j} = p_i$ for all i
  - So expect $\Sigma_i\ p_i\ q_{i+j} \approx 0.065$
- Test for every value of j to find the right one
  - Repeat for each stream

Finding the key length

- When using the correct key length, the ciphertext frequencies $\{q_i\}$ of a stream will be shifted versions of the $\{p_i\}$
  - So $\Sigma\ q_i^2 = \Sigma\ p_i^2 \approx 0.065$
- When using an incorrect key length, expect (heuristically) that the $\{q_i\}$ are equal
  - So $\Sigma\ q_i^2 = \Sigma$ (1/26)^2 = 1/26 = 0.038
- In fact, good enough to find the key length N that maximizes $\Sigma\ q_i^2$
  - Can check with other streams…

Byte-wise Vigenère cipher

- The key is a string of bytes
- The plaintext is a string of bytes
- To encrypt, XOR each character in the plaintext with the next character of the key
  - Wrap around in the key as needed
- Decryption just reverses the process

Example

- Say plaintext is “Hello!” and key is 0xA1 2F
- “Hello!” = 0x48 65 6C 6C 6F 21
- XOR with 0xA1 2F A1 2F A1 2F
- 0x48 ⊕ 0xA1
  - 0100 1000 ⊕ 1010 0001 = 1110 1001 = 0xE9
- Ciphertext: 0xE9 4A CD 43 CE 0E

Attacking the (variant) Vigenère cipher

- Two steps:
  - Determine the key length
  - Determine each byte of the key
- Same principles as before…

Determining the key length

- Let p_i (for $0 \leq i \leq 255$) be the frequency of **byte** i in general English text
  - I.e., $p_i = 0$ for $i < 32$ or $i > 127$
  - I.e., p_{97} = frequency of 'a'
  - The distribution is far from uniform
- If the key length is N, then every N^{th} character of the plaintext is encrypted using the same "shift"
  - If we take every N^{th} character and calculate frequencies, we should get the p_i's in permuted order
  - If we take every M^{th} character (M not a multiple of N) and calculate frequencies, we should get something close to uniform

Vigenère Cipher

- Best known and one of the simplest polyalphabetic substitution ciphers
- In this scheme the set of related monoalphabetic substitution rules consists of the 26 Caesar ciphers with shifts of 0 through 25
- Each cipher is denoted by a key letter which is the ciphertext letter that substitutes for the plaintext letter a

Example

key: deceptivedeceptivedeceptive

plaintext: wearediscoveredsaveyourself

ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Kasiski’s method

TH :	2.71	EN :	1.13	NG :	0.89
HE :	2.33	AT :	1.12	AL :	0.88
IN :	2.03	ED :	1.08	IT :	0.88
ER :	1.78	ND :	1.07	AS :	0.87
AN :	1.61	TO :	1.07	IS :	0.86
RE :	1.41	OR :	1.06	HA :	0.83
ES :	1.32	EA :	1.00	ET :	0.76
ON :	1.32	TI :	0.99	SE :	0.73
ST :	1.25	AR :	0.98	OU :	0.72
NT :	1.17	TE :	0.98	OF :	0.71

THE :	1.81	ERE :	0.31	HES :	0.24
AND :	0.73	TIO :	0.31	VER :	0.24
ING :	0.72	TER :	0.30	HIS :	0.24
ENT :	0.42	EST :	0.28	OFT :	0.22
ION :	0.42	ERS :	0.28	ITH :	0.21
HER :	0.36	ATI :	0.26	FTH :	0.21
FOR :	0.34	HAT :	0.26	STH :	0.21
THA :	0.33	ATE :	0.25	OTH :	0.21
NTH :	0.33	ALL :	0.25	RES :	0.21
INT :	0.32	ETH :	0.24	ONT :	0.20

Kaisiki’s Method

- The first step here is to identify repeated patterns of length 2 or 3 in the ciphertext.

Kaisiki Method

Plaintext:	the man and the woman retrieved the letter from the post office
Key:	bea dsb ead sbe adsbe adsbeadsb ead sbeads bead sbe adsb eadsbe
Ciphertext:	ULE PSO ENG LII WREBR RHLSMEYWE XHH DFXTHJ GVOP LII PRKU SFIADI

Key = beads

The word the is mapped sometimes to ULE, sometimes to LII, and sometimes to XHH. However, it is mapped twice to LII.

Kasiski's observation was that the distance between such repeated appearances must be a multiple of the period.

the period is 5 and the distance between the two appearances of LII is 30, which is 6 times the period

Kaisiki

Consider the ciphertext:

CDGAV	NNANX	DOKVZ	XDGVG	OBMXG	HVLOL
QFZIA	PJAXB	OAGTZ	FBTGA	IBVUK	LOBZT
SMDNV	GKSII	OAGJO	BJLGO	CIDGV	ZZCMH
YFGUI	ZWSBY	VKGUV	FGXFU	ZFOLK	FJOMZ
CMGMO	AGLCC	MWCDG	NCXUW	YCAYG	JALJF
GSXBJ	MJWMC	IECFB	OVZGU	PMOHO	KVHYE
CONNC	XTAQY	MZCJJ	HIXUW	KUMTV	WNNCX
ISPFN	YTGHN	CXCIP	COTPA	OBZFC	JIYVG
FLCYN	XKFZM	ZICJV	NZMJW	HZMHO	LCYWX

- The trigram CDG occurs in positions 1 and 133.
- The bigram DG occurs in positions 2, 17, 83, and 134.
- The bigram NN occurs in positions 6, 183, and 207.
- The trigram OAG occurs in positions 41, 71, and 125.

Kaisiki Method

By assumption, some (but not necessarily all) of these repeated bigrams and trigrams are separated by multiples of k, the keylength:

- The trigram CDG occurs in positions 1 and 133: These are $133 - 1 = 132$ spaces apart.
- The bigram DG occurs in positions 2, 17, 83, and 134: These are $17 - 2 = 15$, $83 - 2 = 81$, $134 - 2 = 132$, $83 - 17 = 66$, $134 - 17 = 117$, and $134 - 83 = 51$ spaces apart.
- The bigram NN occurs in positions 6, 183, and 207: These are $183 - 6 = 177$, $207 - 6 = 201$, and $207 - 183 = 24$ spaces apart.
- The trigram OAG occurs in positions 41, 71, and 125. These are $71 - 41 = 30$, $125 - 41 = 84$, and $125 - 71 = 54$ spaces apart.

If you find every occurrence of every bigram and trigram, you'll generally find . . .

Most of the numbers are divisible by 6, hence key length is 6.
The other digram and trigram are coincidental

Decryption

If we assume a keylength of 6, then every 6th letter comes from the same shift:

C	N	K	V	G
Q	J	G	G	K
S	K	G	G	V
Y	W	G	F	K
C	G	C	U	G
G	J	C	G	O
C	T	C	U	V
I	T	C	P	C
F	K	C	J	O

So the 1st, 7th, 13th, etc., letters are all from the same shift.

Given the high frequency of the ciphertext G, it's reasonable to assume $E \to G$, suggesting a shift 2, and giving the first letter of the keyword: C.

Decryption

If we assume a keylength of 6, then every 6th letter comes from the same shift:

D	A	V	G	H
F	A	T	A	L
M	S	J	O	Z
F	S	U	U	F
M	L	D	W	J
S	W	F	U	K
O	A	J	W	W
S	G	I	A	J
L	F	J	W	L

Next, take the 2nd, 8th, etc. letters,

There are 5 *A*s, 5 *F*s, 5 *J*s and 5 *W*s, so it's harder to tell which might be *E*. So we might try to look at the frequency histograms:

If $E \to W$, then $I \to A$, $N \to F$, $R \to J$, suggesting a plaintext with many *I*s, *N*s, and *R*s.

Index of Coincidence

Letter	Frequency	Letter	Frequency
A	8.17%	N	6.75%
B	1.49%	O	7.51%
C	2.78%	P	1.93%
D	4.25%	Q	0.10%
E	12.70%	R	5.99%
F	2.23%	S	6.33%
G	2.02%	T	9.06%
H	6.09%	U	2.76%
I	6.97%	V	0.98%
J	0.15%	W	2.36%
K	0.77%	X	0.15%
L	4.03%	Y	1.97%
M	2.41%	Z	0.07%

Example

- Consider selecting a pair of letters at random from a very large sample of ordinary English text. The probability that both letters will be As should be as follows.

$$
\begin{aligned}
&P(\text{both letters are A's}) \\
&\quad = \quad P(\text{first letter is an A}) \cdot P(\text{second letter is an A}) \\
&\quad = \quad 0.0817 \cdot 0.0817 \quad = \quad (0.0817)^2 \quad \approx \quad 0.0076
\end{aligned}
$$

Example

$$
\begin{aligned}
&P(\text{any matching pair of letters}) \\
&\quad = \quad P(\text{both are } \texttt{As}) + P(\text{both are } \texttt{Bs}) + \cdots + P(\text{both are } \texttt{Zs}) \\
&\quad = \quad (0.0817)^2 + (0.0149)^2 + \cdots + (0.0007)^2 \quad \approx \quad 0.0655
\end{aligned}
$$

Index of incidence

In order to find the index of coincidence for a given ciphertext, let n_1, n_2, ... , n_{26} be the frequencies with which the letters `A`, `B`, ... , `Z` occur in the ciphertext, respectively, and let $n = n_1 + n_2 + \cdots + n_{26}$ be the total number of letters in the entire ciphertext. If two letters are selected at random without replacement from the ciphertext, then the probability that both letters will be `A`s, `B`s, ... , `Y`s, or `Z`s will be given by the following formulas.

$$P(\text{both letters are A}) = \frac{n_1}{n} \cdot \frac{n_1 - 1}{n - 1} = \frac{n_1(n_1 - 1)}{n(n - 1)}$$

$$P(\text{both letters are B}) = \frac{n_2}{n} \cdot \frac{n_2 - 1}{n - 1} = \frac{n_2(n_2 - 1)}{n(n - 1)}$$

Index of incidence

Monoalphabetic ciphers preserve letter frequencies from plaintexts into ciphertexts. As a result of this, the index of coincidence for a large ciphertext formed using a monoalphabetic cipher should be closer to 0.0655 than 0.0385. Polyalphabetic ciphers typically cause letter frequencies in ciphertexts to be more evenly distributed. Thus, the index of coincidence for a large ciphertext formed using a polyalphabetic cipher should be closer to 0.0385.

$$
\begin{aligned}
& P(\text{any matching pair of letters}) \\
&= P(\text{both are } \texttt{As}) + P(\text{both are } \texttt{Bs}) + \cdots + P(\text{both are } \texttt{Zs}) \\
&= \frac{n_1(n_1-1)}{n(n-1)} + \frac{n_2(n_2-1)}{n(n-1)} + \cdots + \frac{n_{26}(n_{26}-1)}{n(n-1)} \\
&= \frac{1}{n(n-1)}\left(n_1(n_1-1) + n_2(n_2-1) + \cdots + n_{26}(n_{26}-1)\right)
\end{aligned}
$$

Example

Example 7.17 Consider the ciphertext UZRZE GNJEN VLISE XRHLY PYEGT ESBJH JCSBP TGDYF XXBHE EIFTC CHVRK PNHWX PCTUQ TGDJH TBIPR FEMJC NHVTC FSAII IFNRE GSALH XHWZW RZXGT TVWGD HTEYX ISAGQ TCJPR SIAPT UMGZA LHXHH SOHPW CZLBR ZTCBR GHCDI QIKTO AAEFT OPYEG TENRA IALNR XLPCE PYKGP NGPRQ PIAKW XDCBZ XGPDN RWXEI FZXGJ LVOXA JTUEM BLNLQ HGPWV PEQPI AXATY ENVYJ EUEI, which was formed using a Vigenère keyword cipher. There are 269 letters in this ciphertext, and the frequency with which each particular letter occurs is shown in the following table.

Letter:	A	B	C	D	E	F	G	H	I	J	K	L	M
Count:	13	8	12	6	20	7	16	16	15	9	4	10	3
Letter:	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Count:	11	4	18	6	13	8	18	5	7	8	15	8	9

For the index of coincidence for this ciphertext, we have the following.

$$I = \frac{1}{269 \cdot 268}\left((13 \cdot 12) + (8 \cdot 7) + (12 \cdot 11) + \cdots + (9 \cdot 8)\right) \approx 0.0430$$

Since this index of coincidence is closer to 0.0385 than 0.0655, the cipher

Estimating Key Length

$$k \approx \frac{0.0270n}{0.0655 - I + n(I - 0.0385)}$$

Example 7.18 Consider the ciphertext `UZRZE GNJEN VLISE XRHLY PYEGT ESBJH JCSBP TGDYF XXBHE EIFTC CHVRK PNHWX PCTUQ TGDJH TBIPR FEMJC NHVTC FSAII IFNRE GSALH XHWZW RZXGT TVWGD HTEYX ISAGQ TCJPR SIAPT UMGZA LHXHH SOHPW CZLBR ZTCBR GHCDI QIKTO AAEFT OPYEG TENRA IALNR XLPCE PYKGP NGPRQ PIAKW XDCBZ XGPDN RWXEI FZXGJ LVOXA JTUEM BLNLQ HGPWV PEQPI AXATY ENVYJ EUEI`, which was formed using a Vigenère keyword cipher. There are 269 letters in this ciphertext, and in Example 7.17 we found that the index of coincidence for the ciphertext is approximately 0.0430. For an estimate for the length of the keyword for the cipher, we have the following.

$$k \approx \frac{0.0270 \cdot 269}{0.0655 - 0.0430 + 269(0.0430 - 0.0385)} \approx 5.8905.$$