Day 32, Caesar Techniques and Matrix Encryption

We'll present some fairly straightforward encryption techniques individually, but they could be easily combined to make a more challenging encryption scheme.

As with Steganography from last class, many of the early cryptographic codes are attributed to Julius Caesar. The Roman Empire he commanded covered a very large area, and he was concerned that his confidential messages be received without being read.

We'll continue with three of the cryptographic techniques Julius Caesar used:

- Caesar Square – what is the following message? ***h h e i e o t r k***

  Hints: 9 characters, "square," 3 x 3 …

h	i	t
h	e	r
e	o	k

➔ ***hithereok***

  the easiest approach is to fill the square top-bottom, left-right, but then read left-right, top-bottom

- Caesar Cipher – what is the following message?

  104 105 97 108 108

  Hints: convert numeric to alpha, probably easiest to use ASCII, https://www.asciitable.com/

  104 105 97 108 108
  h i a l l

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

- Caesar Cipher with a shift – what is the following message? ***107 104 100 124 111 111***

  Hint: we still convert using the ASCII table, but in this case all the characters are "shifted right" by 3.

  107 104 124 100 111 111
  104 101 121 97 108 108
  h e y a l l

  This can get more complex if the shift "wraps around" a smaller alphabet, and a modular or "rollover" calculation is needed.

Perfect Security is when an attacker intercepts an encrypted message and has no more information about the original message than if they did ***NOT*** have the encrypted message.

One scheme that has perfect security is ***One-Time Pad Encryption***.

Message → ***n*** meaningful bits

Key → ***n*** random bits

Encrypted Message → ***n*** random bits

The conversion has to be invertible. Think of your binary operations, AND, OR, etc. Which could you use to combine the Message and Key to get an Encrypted Message, and then to the Encrypted Message and Key to get the original message?

XOR

Example:

Message →	1 0 1	Encrypted Message →	0 1 1
Key →	1 1 0	Key →	1 1 0
Encrypted Message →	0 1 1	Message →	1 0 1

One-Time Pad Encryption is used in battlefield applications. The "key" is having a random key, and it must be as long as the message. If not, and it was reused, it would be more likely to be cracked.

Continuing this theme of invertibility, what other mathematical structures or elements are invertible, at least sometimes? Functions… relations… ***matrices***!

Remember Matrix-Matrix multiplication:

$$A_{m\,x\,n} * B_{n\,x\,p} = C_{m\,x\,p}$$

Applied to cryptography, we could make ***m***=2 and ***n***=2, and our message, in ASCII and with an even number of characters, could be split between two rows.

Let's use our previous message, ***h e y a l l***, which in ASCII is ***104 101 121 97 108 108***:

$$\begin{pmatrix} 2 & 3 \\ 4 & 1 \end{pmatrix} * \begin{pmatrix} 104 & 101 & 121 \\ 97 & 108 & 108 \end{pmatrix} = \begin{pmatrix} 499 & 526 & 566 \\ 513 & 512 & 592 \end{pmatrix}$$

$$A_{2\,x\,2} \quad * \quad B_{2\,x\,3} \quad = \quad C_{2\,x\,3}$$

The product gives the encrypted message in ***C***, $\begin{pmatrix} 499 & 526 & 566 \\ 513 & 512 & 592 \end{pmatrix}$.

To decrypt, we use the inverse of ***A*** multiplied by ***C*** to recover ***B***:

$$A * B = C$$

$$A^{-1} * A * B = A^{-1} * C$$

$$B = A^{-1} * C$$

$$\begin{pmatrix} 104 & 101 & 121 \\ 97 & 108 & 108 \end{pmatrix} = \begin{pmatrix} -\frac{1}{10} & \frac{3}{10} \\ \frac{4}{10} & -\frac{2}{10} \end{pmatrix} * \begin{pmatrix} 499 & 526 & 566 \\ 513 & 512 & 592 \end{pmatrix}$$

For an in-class lab, work on automating the techniques discussed today: Caesar Square, Caesar Cipher, and Matrix Inversion.

The tools in a Jupyter Notebook are convenient. You are free to solve this however you wish, and you may find the provided Jupyter hints helpful (variable names are in ***bold and italics***).

1. Taking an input string of characters, no spaces, and outputting the Caesar Square decryption.
   You are free to solve this however you wish, you may find the following Python hints helpful (variable names are in ***bold and italics***):
   ```
   import numpy as np
   matrix = np.array( [] )
   matrix = np.append(matrix,char)
   .reshape(rows,cols)
   ```

2. Taking an input string of ASCII numbers, separated by spaces, and outputting the Caesar Cipher decryption.
   You are free to solve this however you wish, you may find the following Python hints helpful (variable names are in ***bold and italics***):
   ```
   .split()
   char_code = int(num)
   chr(char_code)
   ```

3. Taking an input string of characters, separated by spaces, and outputting their corresponding ASCII numbers.
   You are free to solve this however you wish, you may find the following Python hints helpful (variable names are in ***bold and italics***):
   ```
   ord(char)
   ```

4. Take an encrypted message (***C*** in the example above) and the ***2 x 2*** matrix that was used to encrypt it (***A*** in the example above), and recover the original message (***B*** in the example above). :
   ```
   import numpy as np
   np.array(
   C = np.dot( A , B ) # multiplies A and B and puts in C
   Ainv = np.linalg.inv( A )
   ```