Ala-Too International University

College of IT and Business

BASICS OF ALGORITHMIZATION AND PROGRAMM

Activity 0012 - decoding a ciphertext

This task uses a simplified, educational decryption pipeline inspired by a real cryptographic system. Follow the stages in order and show your work.

Ciphertext for this task is **_I54hL9b2JX**

Key (repeat in cycle): **3 3 2 3 3 2 3 3 2 3 3**

Stage 1 Ciphertext symbol decoding (From ciphertext symbols → letter pairs)

Begin the decryption process by mapping each ciphertext symbol to an intermediate letter pair.
A repeating numeric key (composed of 2s and 3s) determines which column of the lookup table must be used for each symbol.
This step simulates the first transformation layer found in real cryptographic pipelines, where encoded symbols are expanded into structured data units before further processing.

Symbol	key=3	key=2
_	BH	ZQ
X	LH	MM
5	YN	LI
J	PH	EE
h	PD	FF
2	HD	AK
9	PP	VR
b	PN	QS
L	YL	PM
4	PL	CQ
I	PE	XR

Stage 2 Letter pair to hexadecimal translation (Letter pairs → hex value)

Each letter pair generated in Stage 1 is translated into a hexadecimal byte using a predefined substitution table. This stage mimics a simplified substitution step similar to operations in classical and modern block ciphers, where symbolic representations must be converted into machine-readable numerical values for subsequent algorithmic processing.

Letter Pair	Hex Value
BH	**E4**
PE	**C1**
LI	**85**
PL	**C8**
PD	**C0**
PM	**C9**
PP	**CC**
PN	**CA**
AK	**D7**
PH	**C4**
LH	**84**
RF	**7E**
BR	**E9**
ND	**B7**
JA	**6F**
CL	**A3**
LE	**A7**
DG	**C3**
OB	**89**
GP	**76**

Stage 3 Inverse S-Box Transformation (XOR Operation)

The hexadecimal values are then passed through a reduced inverse S-Box using a bitwise XOR with the constant A5 (hex).This simulates a core cryptographic primitive used in modern block ciphers such as AES.

Use a simplified inverse S-Box: XOR every byte with A5 (hex).

Use this table to convert any hex digit (0 – F) to its 4-bit binary form.

Example - E4 XOR A5
E4 = E(1110) 4(0100) → **1110 0100**
A5 = A(1010) 5(0101) → **1010 0101**
Apply XOR (bit by bit)

11100100

10100101

01000001

Convert back to hex 0100 0001 = **41**

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Stage 4 ASCII Interpretation

The final stage converts each transformed hexadecimal byte into its corresponding ASCII character. Use the ASCII table to translate result from stage 3 code into its character:

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Stage 5 Java Implementation of the decryption pipeline

The program should:

1. Take the given ciphertext and key sequence as inputs.

2. Use the Stage 1 lookup table to convert each ciphertext symbol into a letter pair, based on the corresponding key digit.

3. Convert each letter pair into its hexadecimal byte using the Stage 2 substitution table.

4. Apply the inverse S-Box step by XOR-ing each byte with the constant A5 (hex), as in Stage 3.

5. Interpret the resulting bytes as ASCII characters to recover the final plaintext message (Stage 4).