Encoding Systems and Symmetric Encryption Lab

Parameters:

A: SimonasRiska

B: 20185536

C: VILNIUSGEDIMINASTECHNIKALUNIVERSITY

Tasks I:

- **A1Z26 code of *SimonasRiska***

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
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

S – 19
I – 9
M – 13
O – 15
N – 14
A – 1
S – 19
R – 18
I – 9
S – 19
K – 11
A – 1

Answer: 19 9 13 15 14 1 19 18 9 19 11 1

- **ASCII code of *SimonasRiska***

ASCII TABLE

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]

S – 83
i – 105
m – 109
o – 111
n – 110
a – 97
s – 115
R – 82
i – 105
s – 115
k – 107
a – 97

Answer: 83 105 109 111 110 97 115 82 105 115 107 97

- **Binary representation of *20185536***

  Since 20185536 divides by 64 ($64 = 2^6$), the number can be simplified by dividing and adding 6 zeros at the binary representation end.

  $$20185536 \div 2^6 = 315399$$

  315399 can be converted to binary but at the end of calculation 6 zeros would need to be appended.

  $$n = 315399$$

  At each step *n* should be divided by 2, remainder should be recorded, *n* updated to the integer quotent of the division and the algorithm should be repeated until *n* equals 0.

n (Before Division)	$n \div 2$ (Quotent)	Remainder
315399	157699	1
157699	78849	1
78849	39424	1
39424	19712	0
19712	9856	0
9856	4928	0
4928	2464	0
2464	1232	0
1232	616	0
616	308	0
308	154	0
154	77	0
77	38	1
38	19	0
19	9	1
9	4	1
4	2	0
2	1	0
1	0	1

  Binary representation of 315399 is 1001101000000000111.
  Since 20185536 was initially divided by 2^6 binary digits of 315399 should be shifted to the left by 6 places, appending 6 zeros at the end.
  Binary representation of 20185536 is 1001101000000000111000000.

  Answer: 1001101000000000111000000

- **Octal representation of *20185536***

  $n = 20185536$
  n should be divided by 8 until $n = 0$ and remainders should be recorded at each step to obtain octal number.

n (Before Division)	$n \div 8$ (Quotent)	Remainder
20185536	2523192	0
2523192	315399	0
315399	39424	7
39424	4928	0
4928	616	0
616	77	0
77	9	5
9	1	1
1	0	1

  Octal representation of 20185536 is 115000700

  Answer: 115000700

- **Hexadecimal representation of *20185536***

  $n = 20185536$
  n should be divided by 16 until $n = 0$ and remainders should be recorded at each step to obtain hexadecimal number.

n (Before Division)	$n \div 16$ (Quotent)	Remainder
20185536	1261596	0
1261596	78849	12
78849	4928	1
4928	308	0
308	19	4
19	1	3
1	0	1

  In hexadecimal 10 – A, 11 – B, 12 – C, 13 – D, 14 – E, 15 – F

  Hexadecimal representation of 20185536 is 13401C0

  Answer: 13401C0

- **Base64 representation of *20185536***

Calculating bytes:

$Byte1 = \lfloor 20185536 \div 256^3 \rfloor = 1$

$20185536 - (1 \times 256^3) = 3408320$

$Byte2 = \lfloor 3408320 \div 256^2 \rfloor = 52$

$3408320 - (52 \times 256^2) = 448$

$Byte3 = \lfloor 448 \div 256 \rfloor = 1$

$448 - (1 \times 256) = 192$

$Byte4 = 192$

Converting each byte to an 8-bit binary number:

$Byte1 = 1 = 00000001$

$Byte2 = 52 = 00110100$

$Byte3 = 1 = 00000001$

$Byte4 = 192 = 11000000$

Combining binary strings:

$Combined\ Binary = 00000001\ 00110100\ 00000001\ 11000000$

Splitting combined binary string into 6-bit groups:

$Group1 = 000000$

$Group2 = 010011$

$Group3 = 010000$

$Group4 = 000001$

$Group5 = 110000$

$Group6 = 00\ (padded\ with\ zeros\ to\ make\ it\ 6 - bit: 000000)$

Converting each group to decimal:

$Group1 = 000000 = 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 0$

$Group2 = 010011 = 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 19$

$Group3 = 010000 = 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 16$

$Group4 = 000001 = 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 1$

$Group5 = 110000 = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 48$

$Group6 = 000000 = 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 0$

BASE64 INDEX TABLE

0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Mapping each value to Base64 table:

$Group1 = 0 = A$

$Group2 = 19 = T$

$Group3 = 16 = Q$

$Group4 = 1 = B$

$Group5 = 48 = w$

$Group6 = 0 = A$

Combining Base64 characters:

$Base64\ String = ATQBwA$

Adding padding characters (because Base64 encoded string should have length that is a multiple of 4 and current string has 6 characters two padding characters "=" is needed to add to make it 8 characters long):

$Final\ Base64\ String = ATQBwA ==$

Answer: ATQBwA==

- **Base58 representation of *20185536***

  $n = 20185536$
  n should be divided by 58 until $n = 0$ and remainders should be recorded at each step to obtain hexadecimal number.

n (Before Division)	$n \div 58$ (Quotent)	Remainder
20185536	348026	28
348026	6000	26
6000	103	26
103	1	45
1	0	1

Value	Character	Value	Character	Value	Character	Value	Character
0	1	1	2	2	3	3	4
4	5	5	6	6	7	7	8
8	9	9	A	10	B	11	C
12	D	13	E	14	F	15	G
16	H	17	J	18	K	19	L
20	M	21	N	22	P	23	Q
24	R	25	S	26	T	27	U
28	V	29	W	30	X	31	Y
32	Z	33	a	34	b	35	c
36	d	37	e	38	f	39	g
40	h	41	i	42	j	43	k
44	m	45	n	46	o	47	p
48	q	49	r	50	s	51	t
52	u	53	v	54	w	55	x
56	y	57	z				

$$[1; 45; 26; 26; 28] = [2; n; T; T; V]$$

$$Base64 = 2nTTV$$

Answer: 2nTTV

Tasks II:

- **Encrypt *SimonasRiska* using Caesar Cipher with key k=20185536 mod26**

$$k = 20185536 - (26 \times \lfloor 20185536 \div 26 \rfloor) = 20$$

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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Values for SimonasRiska:

S	I	M	O	N	A	S	R	I	S	K	A
18	8	12	14	13	0	18	17	8	18	10	0

Applying Caesar Cipher shift:

Letter	Original Value	Shifted Value (+20)	New Value mod 26	New Letter
S	18	18 + 20 = 38	38 – (26 × ⌊38 ÷ 26⌋) = 12	M
I	8	8 + 20 = 28	28 – (26 × ⌊28 ÷ 26⌋) = 2	C
M	12	12 + 20 = 32	32 – (26 × ⌊32 ÷ 26⌋) = 6	G
O	14	14 + 20 = 34	34 – (26 × ⌊34 ÷ 26⌋) = 8	I
N	13	13 + 20 = 33	33 – (26 × ⌊33 ÷ 26⌋) = 7	H
A	0	0 + 20 = 20	20 – (26 × ⌊20 ÷ 26⌋) = 20	U
S	18	18 + 20 = 38	38 – (26 × ⌊38 ÷ 26⌋) = 12	M
R	17	17 + 20 = 37	37 – (26 × ⌊37 ÷ 26⌋) = 11	L
I	8	8 + 20 = 28	28 – (26 × ⌊28 ÷ 26⌋) = 2	C
S	18	18 + 20 = 38	38 – (26 × ⌊38 ÷ 26⌋) = 12	M
K	10	10 + 20 = 30	30 – (26 × ⌊30 ÷ 26⌋) = 4	E
A	0	0 + 20 = 20	20 – (26 × ⌊20 ÷ 26⌋) = 20	U

$Encrypted\ Word = McgihumLcmeu$

Answer: McgihumLcmeu

- **Encrypt *VILNIUSGEDIMINASTECHNIKALUNIVERSITY* using Vigenere cipher with key *SimonasRiska***

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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Match letters length:

P = VILNIUSGEDIMINASTECHNIKALUNIVERSITY (35 characters)
K = SIMONASRISKASIMONASRISKASIMONASRISK (35 characters)

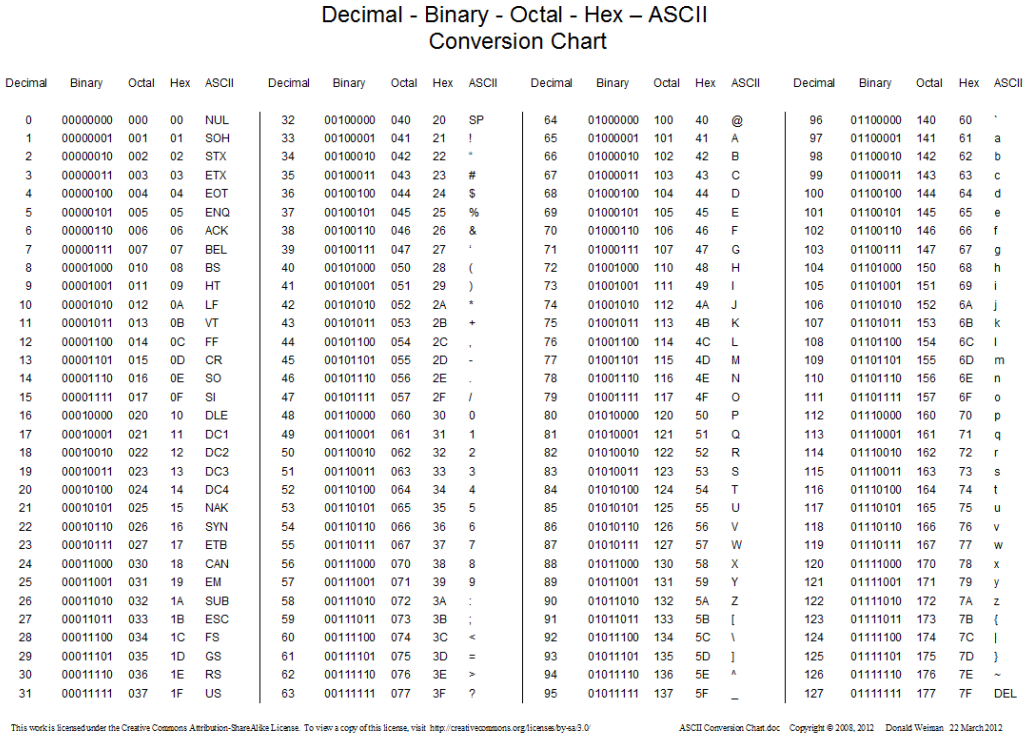
Position	P Letter	P Value	K Letter	K Value	P + K	P + K mod 26	Cipher Letter
1	V	21	S	18	39	$39 - (26 \times \lfloor 39 \div 26 \rfloor) = 13$	N
2	I	8	I	8	16	$16 - (26 \times \lfloor 16 \div 26 \rfloor) = 16$	Q
3	L	11	M	12	23	$23 - (26 \times \lfloor 23 \div 26 \rfloor) = 23$	X
4	N	13	O	14	27	$27 - (26 \times \lfloor 27 \div 26 \rfloor) = 1$	B
5	I	8	N	13	21	$21 - (26 \times \lfloor 21 \div 26 \rfloor) = 21$	V
6	U	20	A	0	20	$20 - (26 \times \lfloor 20 \div 26 \rfloor) = 20$	U
7	S	18	S	18	36	$36 - (26 \times \lfloor 36 \div 26 \rfloor) = 10$	K
8	G	6	R	17	23	$23 - (26 \times \lfloor 23 \div 26 \rfloor) = 23$	X
9	E	4	I	8	12	$12 - (26 \times \lfloor 12 \div 26 \rfloor) = 12$	M
10	D	3	S	18	21	$21 - (26 \times \lfloor 21 \div 26 \rfloor) = 21$	V
11	I	8	K	10	18	$18 - (26 \times \lfloor 18 \div 26 \rfloor) = 18$	S
12	M	12	A	0	12	$12 - (26 \times \lfloor 12 \div 26 \rfloor) = 12$	M
13	I	8	S	18	26	$26 - (26 \times \lfloor 26 \div 26 \rfloor) = 0$	A
14	N	13	I	8	21	$21 - (26 \times \lfloor 21 \div 26 \rfloor) = 21$	V
15	A	0	M	12	12	$12 - (26 \times \lfloor 12 \div 26 \rfloor) = 12$	M
16	S	18	O	14	32	$32 - (26 \times \lfloor 32 \div 26 \rfloor) = 6$	G
17	T	19	N	13	32	$32 - (26 \times \lfloor 32 \div 26 \rfloor) = 6$	G
18	E	4	A	0	4	$4 - (26 \times \lfloor 4 \div 26 \rfloor) = 4$	E
19	C	2	S	18	20	$20 - (26 \times \lfloor 20 \div 26 \rfloor) = 20$	U
20	H	7	R	17	24	$24 - (26 \times \lfloor 24 \div 26 \rfloor) = 24$	Y
21	N	13	I	8	21	$21 - (26 \times \lfloor 21 \div 26 \rfloor) = 21$	V
22	I	8	S	18	26	$26 - (26 \times \lfloor 26 \div 26 \rfloor) = 0$	A
23	K	10	K	10	20	$20 - (26 \times \lfloor 20 \div 26 \rfloor) = 20$	U
24	A	0	A	0	0	$0 - (26 \times \lfloor 0 \div 26 \rfloor) = 0$	A
25	L	11	S	18	29	$29 - (26 \times \lfloor 29 \div 26 \rfloor) = 3$	D
26	U	20	I	8	28	$28 - (26 \times \lfloor 28 \div 26 \rfloor) = 2$	C
27	N	13	M	12	25	$25 - (26 \times \lfloor 25 \div 26 \rfloor) = 25$	Z
28	I	8	O	14	22	$22 - (26 \times \lfloor 22 \div 26 \rfloor) = 22$	W
29	V	21	N	13	34	$34 - (26 \times \lfloor 34 \div 26 \rfloor) = 8$	I
30	E	4	A	0	4	$4 - (26 \times \lfloor 4 \div 26 \rfloor) = 4$	E
31	R	17	S	18	35	$35 - (26 \times \lfloor 35 \div 26 \rfloor) = 9$	J
32	S	18	R	17	35	$35 - (26 \times \lfloor 35 \div 26 \rfloor) = 9$	J
33	I	8	I	8	16	$16 - (26 \times \lfloor 16 \div 26 \rfloor) = 16$	Q
34	T	19	S	18	37	$37 - (26 \times \lfloor 37 \div 26 \rfloor) = 11$	L
35	Y	24	K	10	34	$34 - (26 \times \lfloor 34 \div 26 \rfloor) = 8$	I

Answer: NQXBVUKXMVSMAVMGGEUYVAUADCZWIEJJQLI

Tasks III:

- **Create 64-bits block from *SimonasRiska* (padding or truncating) to use as a plaintext block,**
  create 64-bits block from *20185536* (padding or truncating) to use as a key (real key will be 56-bits) and encrypt using DES

  Since SimonasRiska contains 12 characters (96 bits) and DES operates on 64-bit blocks, the string needs to be truncated to the first 8 characters – SimonasR.



Decimal - Binary - Octal - Hex – ASCII
Conversion Chart

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(	72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29	)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[	123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	|
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D	]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

Table for SimonasR with ASCII codes to each character and their binary representations:

Character	ASCII code	Binary representation
S	83	01010011
i	105	01101001
m	109	01101101
o	111	01101111
n	110	01101110
a	97	01100001
s	115	01110011
R	82	01010010

The key is 20185536, which is 8 characters – 64 bits.

Table for 20185536 with ASCII codes to each character and their binary representations:

Character	ASCII code	Binary representation
2	50	00110010
0	48	00110000
1	49	00110001
8	56	00111000
5	53	00110101
5	53	00110101
3	51	00110011
6	54	00110110

Plaintext “SimonasR” will be encrypted with the key “20185536” using the DES algorithm in ECB mode in Python:

des-task-3.py

C: > Users > simon > Desktop > Cryptography Python > des-task-3.py

```python
from Crypto.Cipher import DES

plaintext = b'SimonasR'
key = b'20185536'

cipher = DES.new(key, DES.MODE_ECB)
ciphertext = cipher.encrypt(plaintext)
print("Ciphertext (hex):", ciphertext.hex())
```

Output:

```
C:\Users\simon\Desktop\Cryptography Python>python des-task-3.py
Ciphertext (hex): 96aea81e4326e172
```

- **Switch one bit in a plaintext block and encrypt with the same key, switch on bit in the key and encrypt the same block,**
  Calculate how many bits changed their value in the ciphertext in both cases

  Switching one bit in a plaintext block:
  Original first byte – 'S' (ASCII 83) – binary 01010011.
  Switched first byte – binary 01010010 – ASCII 82 ('R')
  Modified plain text – "RimonasR"

  Code for switching one bit in a plaintext block and encrypting with the same key:

des-task-3.py

C: > Users > simon > Desktop > Cryptography Python > des-task-3.py

```python
from Crypto.Cipher import DES

plaintext = b'SimonasR'
key = b'20185536'

cipher = DES.new(key, DES.MODE_ECB)
ciphertext = cipher.encrypt(plaintext)

modified_plaintext = bytearray(plaintext)
modified_plaintext[0] ^= 0b00000001
modified_plaintext = bytes(modified_plaintext)

ciphertext_modified_plaintext = cipher.encrypt(modified_plaintext)
print("Ciphertext with Modified Plaintext (hex):", ciphertext_modified_plaintext.hex())
```

  Output:

```
C:\Users\simon\Desktop\Cryptography Python>python des-task-3.py
Ciphertext with Modified Plaintext (hex): c9e2263b126fd02a
```

Switching one bit in a key:
Original first byte – ‘2’ (ASCII 50) – binary 0110010.
Switched first byte – binary 0110011 – ASCII 51 (‘3’)
Modified plain text – “30185536”

Since DES only uses 56 bits for encryption in 64 bits key and first byte is least significant bit (LSB) that serves as a parity bit used for error detection switching on bit in the first key and encrypting the same block would result in obtaining the same ciphertext as it was with the original key.

Code for switching on bit in the key and encrypting the same block:

des-task-3.py

C: > Users > simon > Desktop > Cryptography Python > des-task-3.py

```python
from Crypto.Cipher import DES

plaintext = b'SimonasR'
key = b'20185536'

cipher = DES.new(key, DES.MODE_ECB)
ciphertext = cipher.encrypt(plaintext)

modified_key = bytearray(key)
modified_key[0] ^= 0b00000001
modified_key = bytes(modified_key)

cipher_modified_key = DES.new(modified_key, DES.MODE_ECB)
ciphertext_modified_key = cipher_modified_key.encrypt(plaintext)
print("Ciphertext with Modified Key (hex):", ciphertext_modified_key.hex())
```

Output:

```
C:\Users\simon\Desktop\Cryptography Python>python des-task-3.py
Ciphertext with Modified Key (hex): 96aea81e4326e172
```

I will flip the effective key bit (in this case – bit 1 of the first byte) instead to demonstrate how the ciphertext changes when an effective key bit is modified.

Original first byte – '2' (ASCII 50) – binary 0110010.
Switched first byte – binary 0110000 – ASCII 48 ('0')
Modified plain text – "00185536"

Code for switching bit 1 of the first by in the key and encrypting the same block:

des-task-3.py

C: > Users > simon > Desktop > Cryptography Python > des-task-3.py

```python
from Crypto.Cipher import DES

plaintext = b'SimonasR'
key = b'20185536'

modified_key = bytearray(key)
modified_key[0] ^= 0b00000010
modified_key = bytes(modified_key)

cipher_modified_key = DES.new(modified_key, DES.MODE_ECB)
ciphertext_modified_key = cipher_modified_key.encrypt(plaintext)
print("Ciphertext with Modified Key (hex):", ciphertext_modified_key.hex())
```

Output:

```
C:\Users\simon\Desktop\Cryptography Python>python des-task-3.py
Ciphertext with Modified Key (hex): f282aaa9b8869722
```

Calculating how many bits changed their value in the ciphertext in both cases would be possible with XOR operator and counting differing bits.

Code:

des-task-3.py

C: > Users > simon > Desktop > Cryptography Python > des-task-3.py

```python
from Crypto.Cipher import DES

plaintext = b'SimonasR'
key = b'20185536'

cipher = DES.new(key, DES.MODE_ECB)
ciphertext = cipher.encrypt(plaintext)

modified_plaintext = bytearray(plaintext)
modified_plaintext[0] ^= 0b00000001
modified_plaintext = bytes(modified_plaintext)

ciphertext_modified_plaintext = cipher.encrypt(modified_plaintext)

modified_key = bytearray(key)
modified_key[0] ^= 0b0000010
modified_key = bytes(modified_key)

cipher_modified_key = DES.new(modified_key, DES.MODE_ECB)
ciphertext_modified_key = cipher_modified_key.encrypt(plaintext)

def count_differing_bits(a, b):
    differing_bits = 0
    for byte1, byte2 in zip(a, b):
        xor_result = byte1 ^ byte2
        differing_bits += bin(xor_result).count('1')
    return differing_bits

diff_bits_plaintext = count_differing_bits(ciphertext, ciphertext_modified_plaintext)
diff_bits_key = count_differing_bits(ciphertext, ciphertext_modified_key)

print(f"Bits changed when plaintext is modified: {diff_bits_plaintext}")
print(f"Bits changed when key is modified: {diff_bits_key}")
```

Output:

```
C:\Users\simon\Desktop\Cryptography Python>python des-task-3.py
Bits changed when plaintext is modified: 28
Bits changed when key is modified: 29
```

In an ideal scenario, flipping one bit would change half of the bits in ciphertext, in this case as DES is 64-bit the ideal result should be 32. However, it seems that the number can vary and the 32 is average/approximate value.