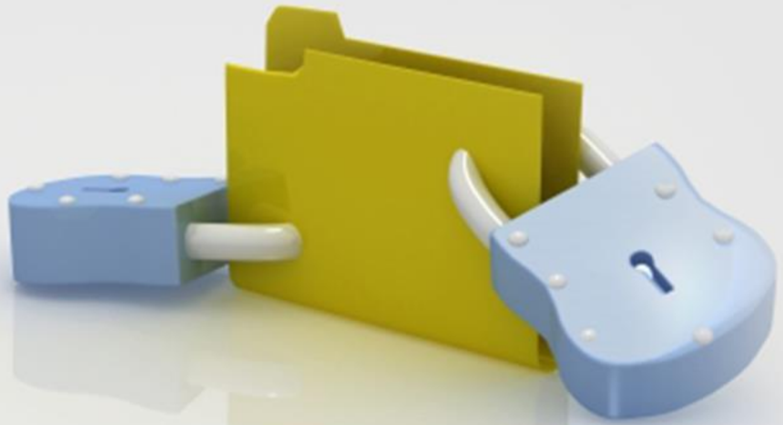
ITEC 111 - Integrated Programming and Technologies 2

ONE-TIME PAD CIPHER Technique

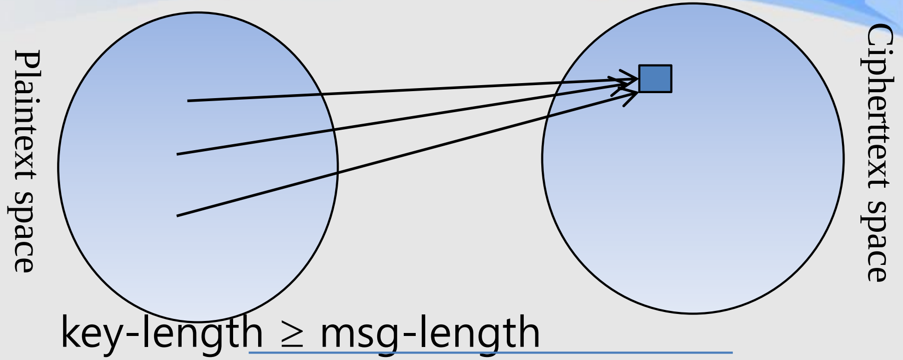
Lecture 5



Truth Excellence Services

One-Time Pad

- type of Vigenere cipher
  - – unbreakable
  - – **Random Key**
  - – **Key should never be reused for any other message to be encrypted**
  - – **Requires key exactly same as the length of message which is encrypted**



Why is it Unbreakable?

- The key is as (long) as the given message.
- The key is truly random and specially auto-generated.
- Key and plain text calculated as modulo 10/26/2.
- Each key should be used once and destroyed by both sender and receiver.
- There should be two copies of key: one with the sender and other with the receiver.

Making Key Length Equal Message Length

Vigenère cipher hacking program works by doing frequency analysis.

If the key is the same length as the message, then every possible ciphertext letter is equally probable to be for the same plaintext letter.

=RANDBETWEEN(1,100)

75	61	84	46	6
94	79	66	69	58
14	92	24	42	8
17	4	82	12	99
75	60	55	45	49
67	46	95	32	36
43	55	97	56	68
47	77	22	72	85
24	67	29	12	11
10	99	31	93	70
37	74	10	66	29
73	54	53	71	32
21	41	5	23	4

Plaintext	ifyouwanttosurviveouthereyouvegottoknowwhereyourtowelis
Key	kcqyzhepxautiqekxejmoretzhztrwwqdylbttvejmedbsanybpxqik
Ciphertext	shomtdecqtilchzssixghyikdfnnmacewrzlghraqqvhzguerplbbqc

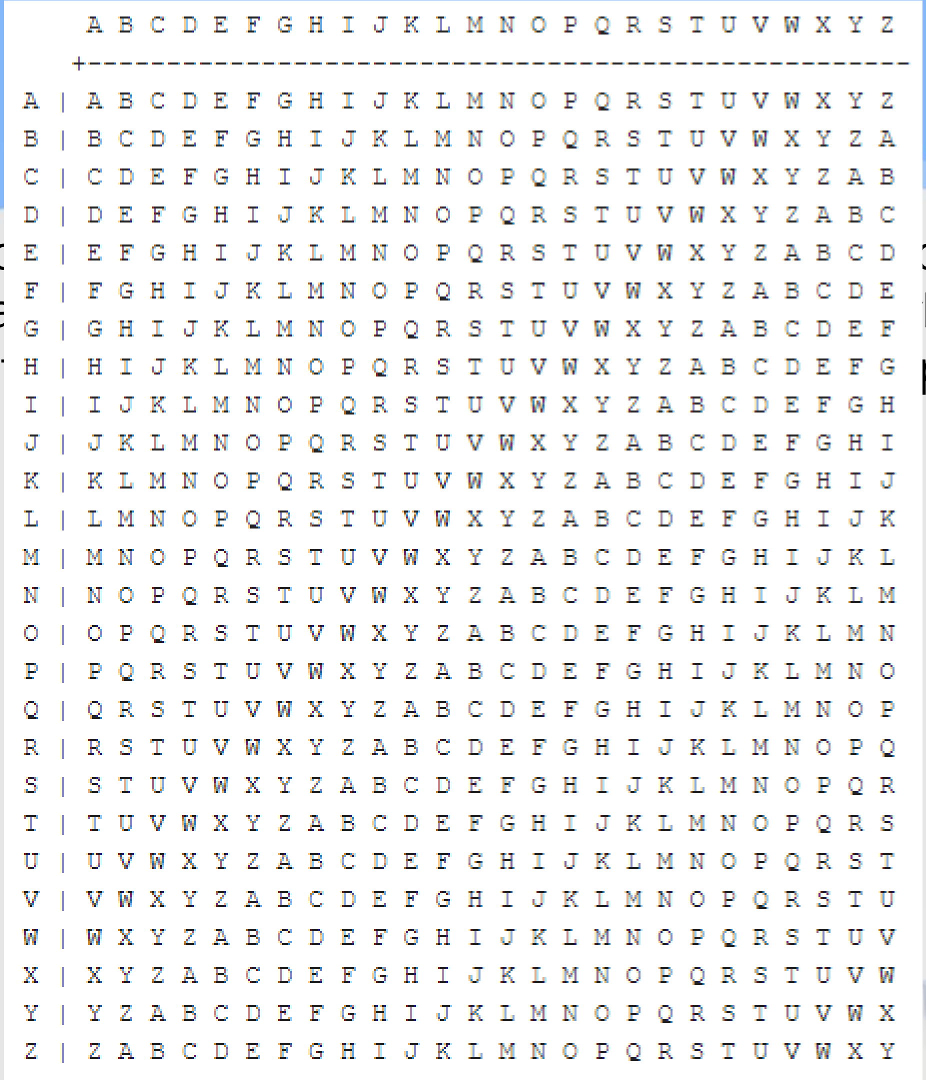
The number of keys would be equal to 26 raised to the power of the total number of letters in the message. So if the message has 55 letters as in the example, there would be a total of 2655, or 666,091,878,431,395,624,153,823, 182, 526,730,590, 376,250,379,528,249,805,353,030,484,209,594, 192,101, 376 possible keys.

This is because for any ciphertext, all possible plaintext messages are equally likely.

Plaintext	ifyouwantt owelis
Key	kcqyzhepxa bpxqik
Ciphertext	shomtdecq plbbqc

… Take the plaintext letter at the top and the key letter on the left. The cross section of those letters is the ciphertext. In the first letter of the example, the crossing between the plaintext **I** and key **K** is ciphertext **S**.

But…



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

Rules of OTP *encryption* with letters:

1. Assign a number to each character of the Plaintext, like (a=0, b=1, c=2, … z=25).
2. Assign a number to each character of the plaintext and the key according to alphabetical order.
3. Add both the number (Corresponding plaintext character number and Key character number).
4. Subtract the number from 26 if the added number is greater than 25, otherwise left it. Assign alphabets of numbers, it produces ciphertext.

Plaintext:	W	A	N	T
	22	0	13	19
Key:	H	E	P	X
	7	4	15	23
SUM (PT + KEY):	29	4	28	42
SUM - 26 (if SUM > 25):	**3**	4	**2**	**16**
Ciphertext:	**D**	**E**	**C**	**Q**

Rules of OTP *decryption* with letters:

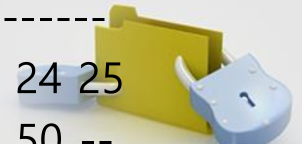
1. Convert the Ciphertext to its equivalent number form
(refer to encryption method step 1)
2. Convert the OTP text to its corresponding numbers
(refer to step 1)
3. Subtract OTP numbers from Cipher Text numbers. If the total is negative, add 26 to it.
4. Convert the Final number to individual alphabets again to get the Final Plain Text

Ciphertext:	**D**	**E**	**C**	**Q**
	3	4	2	16
Key:	H	E	P	X
	7	4	15	23
Diff (CP – Key):	-4	0	-13	-7
Diff + 26 (if DIFF=-):	**22**	0	**13**	**19**
Plaintext:	W	A	N	T

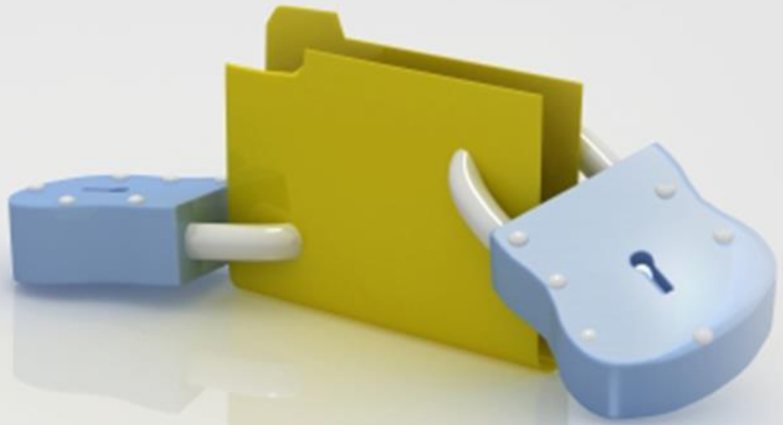
Plaintext :	T	H	I	S	I	S	S	E	C	R	E	T
	19	07	08	18	08	18	18	04	02	17	04	19
OTP-Key:	X	V	H	E	U	W	N	O	P	G	D	Z
	+23	21	07	04	20	22	13	14	15	06	03	25
Result:	42	28	15	22	28	40	31	18	17	23	07	44
Mod 26 =	16	02	15	22	02	14	05	18	17	23	07	18
Ciphertext:	**Q**	**C**	**P**	**W**	**C**	**O**	**F**	**S**	**R**	**X**	**H**	**S**

MODULO 26 HELP TABLE

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
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	--



How the one-time pad cipher works?

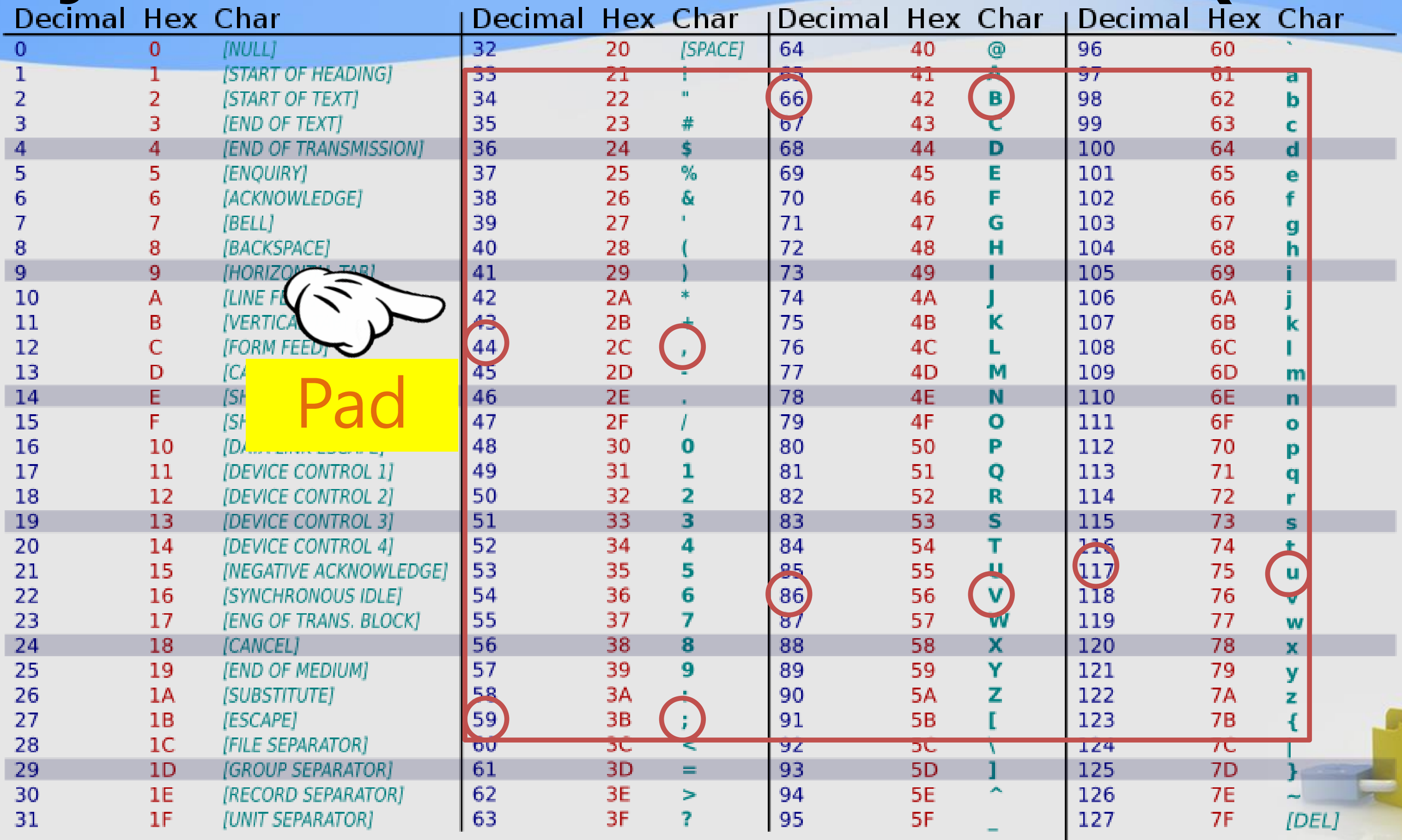


Binary Version of One-Time Pad (ASCII)

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	[illegible]	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	[illegible]	[illegible]	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	[illegible]	[illegible]	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	[illegible]	[illegible]	[illegible]	75	4B	K	107	6B	k
12	C	[FORM FEED]	[illegible]	[illegible]	[illegible]	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	[illegible]	[illegible]	[illegible]	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	[illegible]	[illegible]	[illegible]	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	[ENG 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]

Plaintext

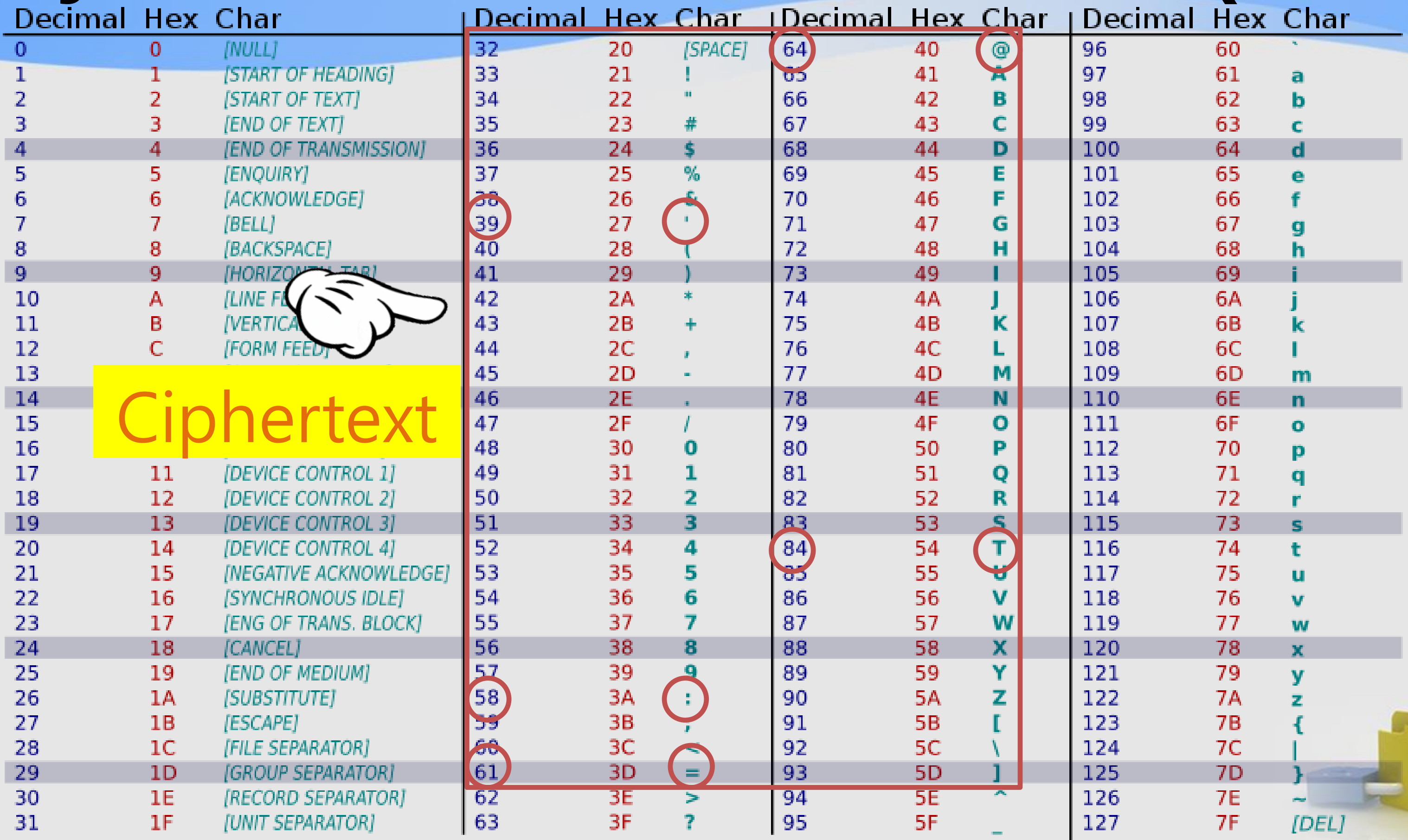
Binary Version of One-Time Pad (ASCII)



Pad

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	[HORIZO[illegible] TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FE[illegible]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICA[illegible]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CA[illegible]	45	2D	-	77	4D	M	109	6D	m
14	E	[SH[illegible]	46	2E	.	78	4E	N	110	6E	n
15	F	[SH[illegible]	47	2F	/	79	4F	O	111	6F	o
16	10	[D[illegible]	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	[ENG 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]

Binary Version of One-Time Pad (ASCII)



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	[HORIZO[illegible]TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FE[illegible]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICA[illegible]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	[illegible]	[illegible]	45	2D	-	77	4D	M	109	6D	m
14	[illegible]	[illegible]	46	2E	.	78	4E	N	110	6E	n
15	[illegible]	[illegible]	47	2F	/	79	4F	O	111	6F	o
16	[illegible]	[illegible]	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	[ENG 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]

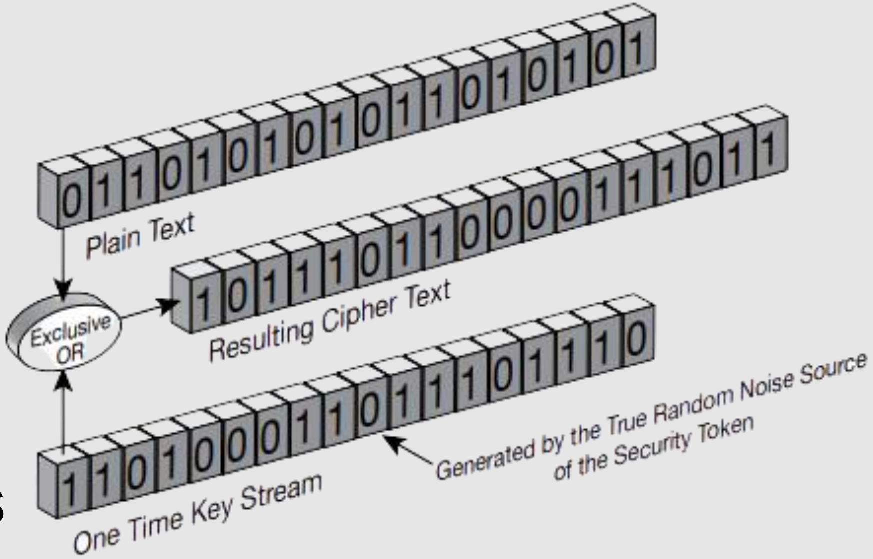
Ciphertext

OTP Algorithm

Both data encryption and decryption by using OTP takes place in the same way. All bytes of the message (or of the ciphertext) are added XOR to bytes of the secret key.

The bytes are added one by one, and each addition produces one output byte:

$$m_i \text{ XOR } k_i = c_i$$
$$c_i \text{ XOR } k_i = m_i$$



Why must we use XOR?

Encryption using random numbers

1. Assign a number to each character of the plain-text, e.g. (a = 0, b = 1, c = 2, … z = 25).

75	61	84	46	6
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
21	41	5	23	4

2. Add plain-text character number to the OTP number.

3. Convert the final number to individual alphabets again to get the FINAL ciphertext.

- **SOLUTIONS:**

Plaintext:	H	e	a	r	t
	7	4	0	17	19
	+	+	+	+	+
OTP:	75	61	84	46	6
=	82	65	84	63	25
	(mod 26)	*(mod 26)*	*(mod 26)*	*(mod 26)*	
	4	13	6	11	25
Ciphertext:	E	N	G	L	Z

Implementation of One Time Pad Cipher

```
# Importing required classes
# Method 1
# Returning encrypted text


def stringEncryption(text, key):
    # Initializing cipherText
    cipherText = ""

    # Initialize cipher array of key length
    # which stores the sum of corresponding no.'s
    # of plainText and key.
    cipher = []
    for i in range(len(key)):
        cipher.append(ord(text[i]) - ord('A') + ord(key[i])-ord('A'))

    # If the sum is greater than 25
    # subtract 26 from it
    # and store that resulting value
    for i in range(len(key)):
        if cipher[i] > 25:
            cipher[i] = cipher[i] - 26

    # Converting the no.'s into integers
    # Convert these integers to corresponding
    # characters and add them up to cipherText

    for i in range(len(key)):
        x = cipher[i] + ord('A')
        cipherText += chr(x)

    # Returning the cipherText
    return cipherText
```

Implementation of One Time Pad Cipher

```
# Method 2
# Returning plain text
def stringDecryption(s, key):

    # Initializing plain text
    plainText = ""

    # Initializing integer array of key length
    # which stores difference
    # of corresponding no.'s of
    # each character of cipherText and key

    plain = []

    # Running for loop for each character
    # subtracting and storing in the array

    for i in range(len(key)):
        plain.append(ord(s[i]) - ord('A') - (ord(key[i]) - ord('A')))

    # If the difference is less than 0
    # add 26 and store it in the array.
    for i in range(len(key)):
        if (plain[i] < 0):
            plain[i] = plain[i] + 26

    # Converting int to corresponding char
    # add them up to plainText

    for i in range(len(key)):
        x = plain[i] + ord('A')
        plainText += chr(x)
    # Returning plainText
    return plainText
```

```
plainText = "Hello"


# Declaring key
key = "MONEY"

# Converting plain text to toUpperCase
# function call to stringEncryption
# with plainText and key as parameters

encryptedText = stringEncryption(plainText.upper(), key.upper())

# Printing cipher Text
print("Cipher Text - " + encryptedText)

# Calling above method to stringDecryption
# with encryptedText and key as parameters
print("Message - " + stringDecryption(encryptedText, key.upper()))    retu
rn plainText
```

THANK YOU

REFERENCE/s:

One Time Pad Cipher. Retrieved September 1, 2021 from https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_one_time_pad_cipher.htm

Hacking Secret Ciphers with Python. Retrieved September 1, 2021 from http://inventwithpython.com/hacking/

Crypto-IT. Retrieved October 3, 2021 from http://www.crypto-it.net/eng/symmetric/otp.html