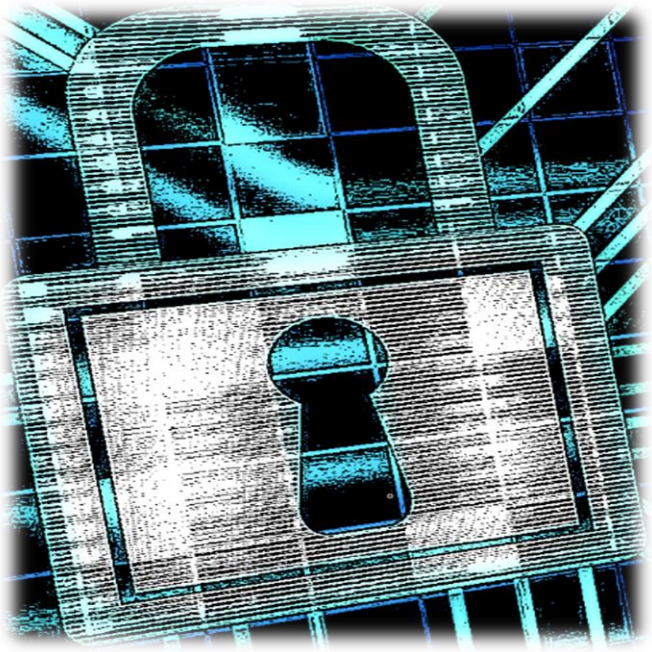
Cyber Security Workshop

Encryption Reference Manual

May 2015



Basic Concepts in Encoding and Encryption

Binary Encoding Examples

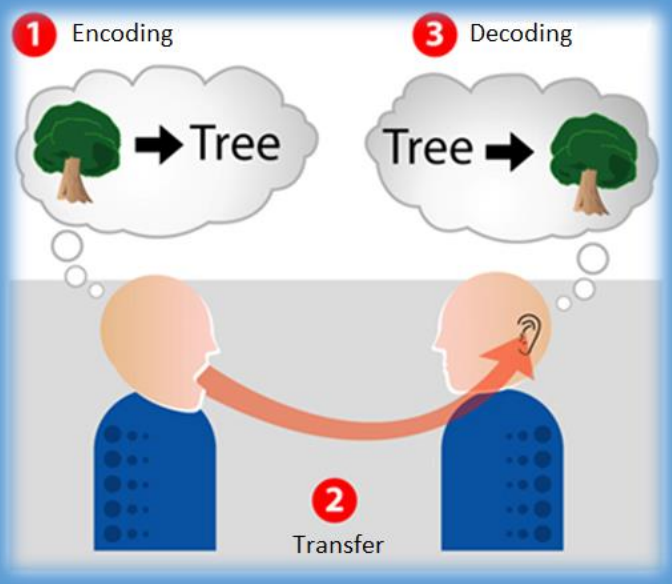
Encryption Cipher Examples

Encoding Concepts

Binary Encoding Basics

Encoding is the art of putting a sequence of characters (such as letters, numbers, punctuations and some symbols) into a specialised digital format for the efficient transfer of data.

Decoding is conversion of an encoded format back into the original sequence of characters.



Binary data is a sequence of 8 binary digits, comprising of 0s and 1s

11000110

ASCII is the most common encoding scheme used in computers and on the internet. ASCII uses 7 binary digits to represent English characters as numbers, with each letter assigned a number from 0 to 127.

Example: Upper-case letter 'A' is represented by the number **65** while lower-case 'z' is represented by the number **122**.

Converting between Decimal (base 10), Binary (base 2) and Hex (Base 16)

A single **Decimal (base 10)** digit can show ten possible values:

0 1 2 3 4 5 6 7 8 9

A single **Binary (base 2)** digit can show two possible values:

0 1

A single **Hexadecimal (base 16)** digit can show sixteen possible values:

0 1 2 3 4 5 6 7 8 9 A B C D E F

Examples:

Ex One - Converting 28 from **Decimal** into **Binary**

1) Draw the following table, filling in the first and third columns:

	2^6	2^5	2^4	2^3	2^2	2^1	2^0
x	0	0	1	1	1	0	0
=	64	32	16	8	4	2	1

2) Compare the number 28 with the numbers in the 3rd row, starting with 64. Our working number is 28.
3) Is 28 less than or equal to 64? Yes, so we place a 0 in the box above 64
4) Is 28 less than or equal to 32? Yes, so we place a 0 in the box above 32
5) Is 28 less than or equal to 16? No, so we place a 1 in the box above 16 and subtract 16 from 28. Our working number is now 12.
6) Is 12 less than or equal to 8? No, so we place a 1 in the box above 8 and subtract 8 from 12. Our working number is now 4.
7) Is 4 less than or equal to 4? Yes, so we place a 1 in the box above 4 and subtract 4 from 4. Our working number is now 0.
8) Once your working number has reached 0, you can then fill in the remaining entries in the second row with 0.

To verify your answer, multiply row 2 with row 3 and add up the entries:
(0 x 64) + (0 x 32) + (1 x 16) + (1 x 8) + (1 x 4) + (0 x 2) + (0 x 1) = 16+8+4 = **28**

Ex Two – Converting 4158 from Decimal into Hexadecimal

1) Draw the following table:

Decimal Number	Operation	Quotient	Remainder	Hex Result
4158	÷ 16 =	**259**	**14**	**E**
259	÷ 16 =	**16**	**3**	**3**E
16	÷ 16 =	**1**	**0**	**0**3E
1	÷ 16 =	0	**1**	**1**03E

2) Divide 4158 by 16 to get a Quotient of 259 and a Remainder of 14.
3) Identify which Hex number matches with decimal 14.
    - Remember: 0 in Hex = 0 decimal, 1 hex = 1 dec, 9 hex = 9 dec
    - 10 in Hex = A , 11 = B, 12 = C , 13 = D, 14 = E, 15 = F
4) Our first hex result is **E** which we determined from our remainder.
5) Divide 259 by 16 to get a new quotient of 16 and a new Remainder of 3.
6) Our second hex result is **3** which we determined from our remainder of 3 (3 in decimal = 3 in hex). Our solution so far is **3E.**
7) Divide 16 by 16 to get a new quotient of 1 and a new Remainder of 0.
8) Our third hex result is 0 (0 in decimal = 0 in hex). Our solution now is **03E**.
9) Divide 1 by 16 to get a new quotient of 0 and a new remainder of 1. Our fourth hex result is 1 (1 in decimal = 1 in hex).

Our final solution now is **103E.** We stop the process once we get a quotient value of 0.

Note: <u>You can also use this technique to convert to binary (Base2)!</u>

Ex Three – Converting Hex to ASCII text using a table

54 68 69 73 20 69 73 20 61 20 6e 6f 74 20 73 6f 20 73 65 63 72 65 74 20 6d 65 73 73 61 67 65 21

An easy method for converting ASCII text to decimal or Hex is by using the table below. Converting the above hex code into ASCII text using the table gives us:

This is a not so secret message!

Note that the number 20 in hex corresponds to the 'SPACE' character in ASCII, and the number 21 in hex corresponds to the '!' character.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	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	Neg. 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 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	Substitution	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	□

Base-64 encoding Overview

Base-64 encoding is a way of taking binary data and turning it into text so that it's more easily transmitted in things like e-mail and HTML form data.

Most computers store binary data in bytes consisting of 8 bits each, so ASCII is not ideal for transferring this type of data.

Base-64 encoding uses 64 characters to encode strings, as seen in the table below:

Value	Char	Value	Char	Value	Char	Value	Char
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	/

1) A-Z, a-z, 0-9, +, and / are the 64 real characters
2) The '=' sign is the padding character, which we add to the end of the string to make it a multiple of 4

During the exchange of information over the internet, on web servers data is converted between Base-64 to ASCII and back to optimise efficiency. So it is therefore important that we gain a basic understanding on how to convert between ASCII and Base-64.

Encode the string 'food' using Base-64

- Convert the text food to 8-bit binary.
  - We can do this by looking up the letters 'f', 'o', 'o' and 'd' on the ASCII table on page 5, find the decimal equivalent value and convert the decimal value to 8-bit binary.
    Note: If your binary result is 7-bit (e.g. 1100110 for the letter 'f', just add an extra 0 to get 8-bit (e.g. 01100110 for 'f')

ASCII Text	f	o	o	d
Decimal	102	111	111	100
Binary	01100110	01101111	01101111	01100100

- Convert the text to 6-bit binary
  - Put all the binary numbers together like so:
    01100110011011110110111101100100
  - Then, break them apart into groups of six, like so:
    011001 100110 111101 101111 011001 00
  - We see that in the final column, we just have two characters, so we pad it out by adding four more zeros to give:
    011001 100110 111101 101111 011001 000000

- Convert the 6-bit binary back into decimal:

011001	100110	111101	101111	011001	000000
25	38	61	47	25	0

- Convert the decimal numbers to ASCII:

25	38	61	47	25	0
Z	m	9	v	Z	A

- The Zm9vZA string is six characters long. Since six is not a multiple of four, we need to add padding characters to increase the total number of characters to eight (which is a multiple of four).
  - Therefore we add two '=' characters to get eight characters: Zm9vZA== which is food in base-64

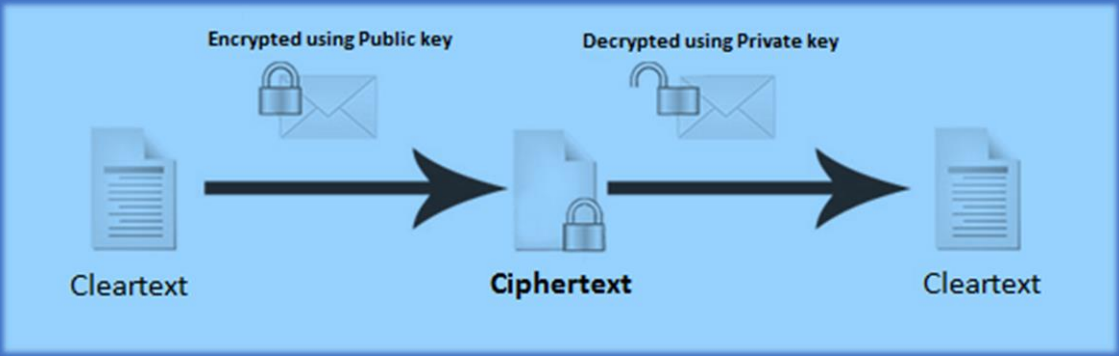
Encryption Basics

Encryption is the art of converting a plain message into a form called cipher text, which cannot be easily understood by unauthorised people. The cipher text can only be viewed in its original form if decrypted with the correct key.

Data files and communications are often encrypted to protect the confidentiality and integrity of the data, to minimise the risk of tampering.

Sometimes, we want to encrypt data so that only a certain person is able to decrypt and read it. We can do this using Public-Private key pairs. For example:

- Bob wants to send an encrypted message to Jane
- Jane sends Bob her **Public Key**, a key that can be used by Bob to encrypt the message specifically for Jane.
- Bob then sends the encrypted message to Jane
- Jane uses her **Private Ke**y, a key that only she has access to, in order to decrypt and read the message.



In cryptography, a **cipher** is an algorithm or a method used for performing encryption or decryption. Different ciphers result in different cipher text, where some types are harder to crack than others. In the next few sections you will be introduced to two basic ciphers – Transposition and Caesar.

Transposition Cipher Example

The Transposition cipher works by messing with the order of the letters to hide the message being sent. In this example, we will look at Columnar Transposition, where the 'encryption key' specifies the number of columns used in the rectangular array.

Exercise – Encrypt the message 'ATTACK AT DAWN FROM WEST' using the transposition cipher with key set to 5.

To apply this encryption, we create a table with 5 columns (corresponding with the key) and populate it with the letters from left to right, like so:

A	T	T	A	C
K	A	T	D	A
W	N	F	R	O
M	W	E	S	T

Next, we record each column as a word, resulting in the following cipher text:

AKWM TANW TTFE ADRS CAOT

If the key for the transposition cipher was set to 3, then we would have obtained the following cipher text instead:

AAAAFMS TCTWRWT TKDNOE

Generally it is easy to reverse cipher text encrypted using the transposition cipher, as all you need to do is look at the number of words and set that as the number of columns in the table. Then you can populate the table accordingly, and obtain the cleartext.

Caesar Cipher Example

The Caesar cipher (also known as the rotation or shift cipher) is a type of substitution cipher where each letter is 'shifted' a certain number of places down the alphabet.

For example, with a shift (key) of 1, the letter A would be replaced by B, letter B would become C and so on. With a shift of 25, the letter A would be replaced by Z, letter B would become A and so on.

Note that the maximum shift is 25. As we only have 26 letters in the alphabet, we can only shift the letters 25 times.

Exercise 1 – Encrypt the message DEFEND THE FRONT OF CASTLE using a Caesar Cipher with a shift of 1.

Plaintext	**DEFEND**	**THE**	**FRONT**	**GATE**	**OF**	**CASTLE**
Ciphertext	EFGFOE	UIF	GSPOU	HBUF	PG	DBTUMF

Exercise 2 – Decrypt the message 'tmhsdb bxadq rdbtqhsx kza'

In order to decrypt this message, we need match the common letters in the Ciphertext with common letters in the English language.

We can see that the letters 'b' and 'd' are the most common letters, with 3 instances each. The four most common letters in the English language are: E (12.7%) , T (9.1%), A (8.2%) and O (7.5%).

So let's start by doing some guess work, by matching the two sets of letter together to see if we get something. Let's assume that 'b' is the letter 'e' in the English language. This would then imply that a shift of 23 has been applied to the original text. If we apply a reverse shift of 23, we get:

wpkvge eadgt ugewtkva ncd

It looks our guess was not right. Let's now try matching the letter 'd' with the English letter 'e', giving us a reverse shift of 25 to apply on the text:

unitec cyber security lab

That looks readable now. We were able to crack the code!