Hexafid Field Cipher v0.7.3 - Cipher Block Chaining (CBC) Mode

A post-apocalyptic field cipher combining classical and digital constructs

Objective: Design a cipher that works easily with pen and paper, secures confidentiality of information, offers plausible deniability if discovered, and exhibits greater strength implemented in software

Note: CBC Mode can only be semantically secure beyond one block with a random Initialization Vector (IV); Recommended use is only for short messages with short lifespans and rotated unique keys

Key Square

M	y	P	a	s	w	o	r
d	1	2	3	A	B	C	D
E	F	G	H	I	J	K	L
N	O	Q	R	S	T	U	V
W	X	Y	Z	b	c	e	f
g	h	i	j	k	l	m	n
p	q	t	u	v	x	z	0
4	5	6	7	8	9	+	/

Ks

1	..
..	64

Kq

00	01
11	10

Sequence Addresses

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	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

e.g. K = 23

Ks

1	..
..	64

Kq

00	01
11	10

Quadtree Addresses

00	00	00	00	01	01	01	01
00	00	00	00	01	01	01	01
00	00	00	00	01	01	01	01
00	00	00	00	01	01	01	01
11	11	11	11	10	10	10	10
11	11	11	11	10	10	10	10
11	11	11	11	10	10	10	10
11	11	11	11	10	10	10	10

e.g. K = 011000

Original: Keep it secret! Keep it safe!

Plaintext: Keep/it/secret/Keep/it/safe

IV: w3Der4LeOd <- a set of random characters equal in length to block size (10)

	Block 1										Block 2										Block n										
	K	**e**	**e**	**p**	**/**	**i**	**t**	**/**	**s**	**e**		**c**	**r**	**e**	**t**	**/**	**K**	**e**	**e**	**p**	**/**		**i**	**t**	**/**	**s**	**a**	**f**	**e**	**P**	**P**	**P**	<- Plaintext
	23	39	39	49	64	43	51	64	5	39		38	8	39	51	64	23	39	39	49	64		43	51	64	5	4	40	39	3	3	3	
	w	**3**	**D**	**e**	**r**	**4**	**L**	**e**	**O**	**d**	->	**L**	**C**	**I**	**8**	**1**	**k**	**j**	**1**	**u**	**7**	->	**v**	**J**	**c**	**T**	**F**	**f**	**B**	**t**	**s**	**s**	
	6	12	16	39	8	57	24	39	26	9		24	15	21	61	10	45	44	10	52	60		53	22	38	30	18	40	14	51	5	5	
	29	51	55	24	8	36	11	39	31	48		62	23	60	48	10	4	19	49	37	60		32	9	38	35	22	16	53	54	8	8	
	S	**t**	**z**	**L**	**r**	**Z**	**2**	**e**	**U**	**n**		**9**	**K**	**7**	**n**	**1**	**a**	**G**	**p**	**b**	**7**		**V**	**d**	**c**	**Y**	**J**	**D**	**v**	**x**	**r**	**r**	<- Intertext
	0	1	1	0	0	1	0	1	0	1		1	0	1	1	0	0	0	1	1	1		0	0	1	1	0	0	1	1	0	0	
	1	1	0	1	1	1	0	0	1	0		0	1	1	0	0	0	0	1	0	1		1	0	0	1	1	1	0	0	1	1	
	1	1	1	1	0	0	0	0	1	0		1	1	1	0	0	0	1	1	0	1		1	0	1	0	1	0	1	1	0	0	
	1	0	0	0	1	1	1	1	0	1		1	0	0	1	0	1	0	1	0	0		0	0	0	1	1	1	1	1	1	1	
	1	0	0	0	0	0	1	0	1	1		1	0	1	1	1	0	0	0	0	0		1	1	0	0	0	1	0	0	0	0	
	1	0	0	1	1	1	1	0	1	0		0	0	0	0	0	1	0	0	0	0		0	1	1	0	1	0	0	1	1	1	
	L	**C**	**I**	**8**	**1**	**k**	**j**	**1**	**u**	**7**	->	**v**	**J**	**c**	**T**	**F**	**f**	**B**	**t**	**s**	**s**	->	**E**	**h**	**I**	**t**	**v**	**2**	**4**	**P**	**3**	**m**	<- Ciphertext
	0	1	1	0	0	1	0	1	0	1		1	0	1	1	0	0	0	1	1	1		0	0	1	1	0	0	1	1	0	0	
	1	1	0	1	1	1	0	0	1	0		0	1	1	0	0	0	0	1	0	1		1	0	0	1	1	1	0	0	1	1	
	1	1	1	1	0	0	0	0	1	0		1	1	1	0	0	0	1	1	0	1		1	0	1	0	1	0	1	1	0	0	
	1	0	0	0	1	1	1	1	0	1		1	0	0	1	0	1	0	1	0	0		1	0	1	1	1	0	0	0	1	0	
	1	0	0	1	1	1	1	0	1	0		0	0	0	0	0	1	0	0	0	0		0	1	1	0	1	0	0	1	1	1	
	S	**t**	**z**	**L**	**r**	**Z**	**2**	**e**	**U**	**n**		**9**	**K**	**7**	**n**	**1**	**a**	**G**	**p**	**b**	**7**		**V**	**d**	**c**	**Y**	**J**	**D**	**v**	**x**	**r**	**r**	<- Intertext
	29	51	55	24	8	36	11	39	31	48		62	23	60	48	10	4	19	49	37	60		32	9	38	35	22	16	53	54	8	8	
	w	**3**	**D**	**e**	**r**	**4**	**L**	**e**	**O**	**d**	->	**L**	**C**	**I**	**8**	**1**	**k**	**j**	**1**	**u**	**7**	->	**v**	**J**	**c**	**T**	**F**	**f**	**B**	**t**	**s**	**s**	
	6	12	16	39	8	57	24	39	26	9		24	15	21	61	10	45	44	10	52	60		53	22	38	30	18	40	14	51	5	5	
	23	39	39	49	0	43	51	0	5	39		38	8	39	51	0	23	39	39	49	0		43	51	0	5	4	40	39	3	3	3	
	K	**e**	**e**	**p**	**/**	**i**	**t**	**/**	**s**	**e**		**c**	**r**	**e**	**t**	**/**	**K**	**e**	**e**	**p**	**/**		**i**	**t**	**/**	**s**	**a**	**f**	**e**	**P**	**P**	**P**	<- Plaintext

Block Padded Ciphertext: LCI81kj1u7vJcTFfBtssEhItv24P3m <- with encrypted pad (Step 6d.)

Embedded IV Ciphertext: w3Der4LeOdLCI81kj1u7vJcTFfBtssEhItv24P3m <- IV is prepended as first block

B64 Padded Ciphertext: w3Der4LeOdLCI81kj1u7vJcTFfBtssEhItv24P3m <- no Base64 pad here (Step 11a.)

Influences and References

Classical ciphers - Bifid/Trifid - Delastelle 1902 (for fractionated substitution)

Information Therory - A Mathematical Theory of Communication - Shannon 1948 (for bits and entropy)

Quadtree Data Structures - Finkel and Bentley 1974 (for key square addressing)

Six Bit Character Encoding - PEM Standard 1987 (for the origins of Base64 character set) and RFC 4648

Cipher Block Chaining - Message verification and transmission error detection by block chaining - Ersham et al. 1976

Initial Concept: 2020-02-25

Latest Update: 2020-08-08

Key Setup

Step 1. Draw an 8x8 key square with 64 cells

Step 2. Populate the key square randomly and uniquely with the characters A-Z a-z 0-9 +/

Key Square Lookup

Step 3. Reference a quadtree address (i.e. quadress) for each character in the keysquare (e.g. 010101)

- a. The first 2 bits represent one of the top key square quadrants addressed clockwise as 00,01,10,11 then
- b. The second 2 bits represent the middle subdivision of that quadrant into fours, addressed the same way; and
- c. The third 2 bits represent a final bottom subdivision, in the same way; arriving at a 6 bit binary number

Step 4. Reference a sequence address (e.g. 1..64) for each character in the key square

- a. Each character in the keysquare is given a consecutive sequence number, cell by cell then row by row

Message Preparation

Step 5. Remove non-key characters from the message

- a. Note: With no space in the character set, message words can be joined together or separated by /

Plaintext Encryption

Step 6. Separate plaintext into blocks of period size and add block padding at end (see below)

- a. Note: the plaintext, inclusive of the IV, must be padded to a period multiple AND mod 4 != 1

Step 7. Using modular addition (64), add current plaintext block to previous ciphertext block (or IV for first block)

- a. Note: use each character's keysquare sequence address for the modular addition

R Step 8. Substitute resulting intertext characters into matching quadress using key square

O - a. Note: written vertically in columns underneath each character

U Step 9. Transpose substituted quadress within each block

N - a. Note: read horizontally in rows, 6 bits at a time within one block at a time

D Step 10. Decode transposed quadress into ciphertext characters using key square

Step 11. Assemble blocks together again as ciphertext; prepend with IV and then add Base64 padding (see below)

- a. Note: only encryption creates an IV; by prependeding to ciphertext, it becomes available for decryption

Ciphertext Decryption

Step 6. Remove any non-key characters and separate ciphertext into blocks of period size

- a. Separate the IV from the ciphertext (ie it was prepended as first block)

R Step 7. Substitute ciphertext characters into matching quadress using key square

O - a. Note: written horizontally in rows, 6 bits at a time within one block at a time

U Step 8. Transpose substituted quadress within each block

N - a. Note: read vertically in columns underneath each character

D Step 9. Decode transposed quadress into intertext characters using key square

Step 10. Using modular subtraction (64), subtract previous ciphertext block (or IV) from current intertext block

- a. Note: use each character's keysquare sequence address for the modular subtraction

Step 11. Assemble blocks together again and remove padding equal to sequence number of last character

Block Padding - The plaintext should be a multiple of the block size

Step 6. (Encryption) Add padding to end of plaintext before encryption

- a. Determine pad length by subtracting length of last block from block size
- b. If pad length is zero, add a full block of padding
- c. Choose pad character from key square at sequential position equal to pad length
- d. Add pad length of pad characters to end of plaintext message

Base 64 Padding - The ciphertext should appear encoded as Base64

Step 11. (Encryption) Add padding to full ciphertext after encryption

- a. If len(ciphertext) mod 4 = 0 then pad nothing
- b. If len(ciphertext) mod 4 = 1 then review Step 6a. (i.e. should not occur)
- c. If len(ciphertext) mod 4 = 2 then pad ==
- d. If len(ciphertext) mod 4 = 3 then pad =