Encryption and security: the Data Encryption Standard

Many attempts have been made to crack the DES (Data Encryption Standard) since the late 1970s, but it has mostly been inexpensive computing power and brute-force methods that have weakened the standard. Still, DES remains the de facto algorithm for encryption.

By Stuart Allman, Cypress Semiconductor

Digital data security is a hot topic in network and storage engineering. Most people can rely on the fact that there is so much information that no one is willing to spend the time or the effort to sort through even a small fraction of a digital data stream. However, in certain applications, it is desirable, if not mandatory, to encrypt the data passing from one device to another. One common and proven method of encryption is DES (Data Encryption Standard).

Work on DES began in the mid 1970s at IBM. At that time, the NBS (National Bureau of Standards, now the National Institute of Standards and Technology, or NIST) requested an efficient algorithm for encryption. Unlike previous efforts, the details of its implementation would be made public, and the NBS would encourage the public to break it. After many public reviews and comments from the National Security Agency, the standard was approved in 1977.

Since that time, DES has been used for midlevel security applications, such as unclassified government information and financial transactions. Modern technology has made the speed and processing requirements for DES incredibly inexpensive and easy to implement, even for low-cost applications. Two relatively new uses for DES are in common set-top boxes and inexpensive secure flash cards. With the proliferation of DES into our common lives, it is important for the engineering community and information professionals to understand what DES is and does.

What is DES?

DES is a method to encrypt a 64-bit block of data using a 56-bit key. The result of this operation is a 64-bit chunk of encrypted data. Some documentation indicates that DES uses a 64-bit key, which is (sort of) true. Eight bits of the 64-bit key value are simply parity bits that the encryption algorithm never uses. Due to modern cryptanalysis techniques, the real cipher strength of DES is 46 to 47 bits, not 56 bits. (See **references 1** through **5**, for more information.)

The most common implementation of DES is the ECB (Electronic Code Book) mode. Other variations of the algorithm exist, some of which **references 1** and **2** describe. Most of these variations are simple manipulations of the same algorithm. ECB-mode implementation is the most commonly deployed DES mode.

DES loops through the main part of the algorithm 16 times. Each of these loops is called a DES “round.” Each round comprises simple bit manipulations and look-up table operations. A round uses a new key derived from the original 56-bit key and produces a new 64-bit data block. The data block at the end of the 16 rounds is the DES encrypted data.

The DES algorithm is symmetric, meaning that the same algorithm both encrypts and decrypts the data. The difference between these two operations is the order in which the algorithm uses the 16 derived keys. The algorithm reverses the key order for the decryption operation.

DES terminology

DES references use a number of terms that you should become familiar with before exploring the algorithm details.

- A *round* is an iteration of the main part of the DES algorithms. The DES algorithm comprises a total of 16 rounds.

- *Permutation* is a reordering of the bits in a chunk of data. Sometimes, the operation involves dropping bits (compression) or repeating bits (expansion) from the starting data.
- *Ciphertext* is the encrypted data.
- *Plaintext* is the original unencrypted data
- Each DES round receives a new key from the *key schedule*. The 16 keys are derived from rotation and permutation operations on the 56-bit DES key. The algorithm requires this computation only once, at initialization.
- An *S-box* is a look-up table.
- In the DES standard, bit ordering starts at bit 1 and increments up, as opposed to starting at bit 0. *Bit 1* refers to the MSB (most significant bit) in a data chunk of arbitrary bit length.

Deriving the key schedule

As previously stated, 16 keys are derived from the original 56-bit key. The key starts out as a 64-bit value. Every eighth bit is an odd parity bit. You may remove and throw away the parity bits once you verify the key to be a valid value.

Before the first round, the DES key derivation algorithm puts the 56-bit key through a key permutation. This operation both rearranges the bits in the key and splits the key into two 28-bit halves (**Figure 1**).

Tables 1 and **2** show the bit reassignments for the left (C_0) and right halves (D_0). In **Table 1**, bit 57 of the key becomes the MSB of the left-half key (bit 1), and so forth until bit 28.

After the initial key permutation, both halves rotate left during each of the 16 iterations, then go through a compression permutation to produce 16 keys (one for each DES round). When deriving the key for the first, second, ninth, and 16th round, the left and right halves rotate left through one bit position. For all other rounds, the halves rotate left through two bit positions. **Figure 2** shows the key derivations operation. Note that the two 28-bit halves recombine into a single 56-bit quantity before going through the compression permutation (**Table 3**). The DES key derivation algorithm must compute the key schedule only once, at initialization. The DES encryption algorithm uses the same key schedule for all succeeding DES cipher operations.

Data initialization

To be DES-compliant, the DES encryption algorithm first puts the 64 bits of plaintext through an initial permutation. Many implementations of DES skip this step, as well as the final permutation. These two steps are simply the inverse of each other and are not necessary for encryption purposes. Many people believe these steps were put in to simplify byte-wide data loads into the original hardware DES implementations and have no modern purpose. However, for completeness, it is helpful to become familiar with the initial permutation operation. **Figure 3** shows the initial permutation operation, and **Table 4** shows the bit reordering.

DES rounds

The DES round is the core of the DES algorithm. The 64-bit plaintext, post-initial permutation goes through a DES round 16 times during the encryption processing. **Figure 4** shows the DES round. During a DES round, the algorithm puts the right-half data (R_{n-1}) through an expansion permutation, which converts the 32-bit data value to a 48-bit expanded value. See **Table 5** for the bit-permutation assignments.

The result of the expansion permutation goes through an exclusive-OR operation with the scheduled key for the DES round. The algorithm separates the 48-bit result of the exclusive-OR operation into eight 6-bit fields. (Field 1 equals bits 1 to 6, field 2 equals bits 7 to 12, and so on.) The algorithm then puts each 6-bit field into one of the eight S-boxes, as **Figure 4** shows. Each S-box produces a 4-bit result, which **tables 6** through **13** describe. In practice, each of the S-boxes is a

simple LUT (look-up-table) operation. Each S-box LUT entry is addressed by separating the first and last bits (S_1S_6 in **Table 6**) and the middle bits ($S_{2...5}$ in **Table 6**) into two indexes.

Post S-box, the DES algorithm inputs the 48-bit output of the S-box LUT operation into a compression permutation, called the "P" permutation. The output of this permutation is 32-bits long. **Table 14** gives the permutation mapping.

Post "P" permutation, the DES algorithm puts the output of the P permutation through an exclusive-OR operation with the left half of the data that you started with (L_{n-1}). The result of the "P" and L_{n-1} exclusive-OR goes into the right-half data (R_n), except in round 16, when the output stays in the left half (L_n). The right half from the previous round (R_{n-1}) is written into the left-half output (L_n), except in round 16, when the right half remains where it is.

The final data manipulation is the inverse permutation (IP^{-1}) of the IP (initial permutation). As previously stated, this step has no cryptographic value. Many software implementations skip this value if they also skip the IP. **Figure 5** and **Table 15** show how the final permutation moves bits from the left and right data halves into the final output

Deciphering DES

The DES algorithm is symmetric, meaning that it uses the previously explained method also to decrypt cipher text. However, in doing so, it introduces the key schedule backward. During deciphering, the DES algorithm uses key 16 in place of key 1, key 15 in place of key 2, and so on, until it uses key 1 in place of key 16. No other changes are necessary to decipher DES-encoded ciphertext.

Triple DES

Because of the ever-increasing horsepower of modern computing, many implementations have turned to triple-DES to ensure security. Triple-DES involves nothing more than running DES three times with three 56-bit keys. However, the second operation is a DES-deciphering operation; the first and third operations are standard DES ciphering. **Figure 6** shows the operation of triple DES.

The method to decipher triple-DES first involves deciphering the cipher text with the third key, ciphering it with the second key, and then deciphering it with the first key. The cipher strength of triple DES is approximately 146 bits with modern cryptography attacks, so it is much stronger than standard ECB DES and clearly infeasible to break with most contemporary technology.

Other modes of DES

Other than ECB mode, the NIST has standardized the CBC (Cipher Block Chaining), CFB (Cipher Feedback), and OFB (Output Feedback) modes. The CBC mode is a method for protecting against fraudulent data insertion into the data stream (**Figure 7**). The CBC mode of operation divides the data stream to be ciphered into 64-bit data chunks. The first data chunk goes through an exclusive-OR with a user-chosen 64-bit initialization vector. The output of the exclusive-OR then goes through an ECB-DES ciphering operation. When it comes time to cipher the next 64 bits using the CBC mode, the CBC-DES algorithm uses the output of the previous ciphering operation in place of the initialization vector. Thus, you cannot insert data into the data stream without agreeing to change the initialization vector and re-ciphering the entire sequence.

CFB is a more complicated mode of DES (**Figure 8**). The cipher starts out by ciphering a user-chosen 64-bit initialization vector using ECB DES. The output of the DES cipher then goes through an exclusive-OR with the first chunk of plain text. The output of the exclusive-OR is the first chunk of cipher text. To cipher the following chunks, the CFB-DES algorithm uses the cipher text from the previous operation in place of the initialization vector.

OFB is a slightly modified CFB mode (**Figure 9**). The cipher operation starts by ciphering a user-chosen 64-bit initialization vector using ECB DES. The output of the DES operation goes

through an exclusive-OR with the plain text to form the first cipher text. During the next ciphering operation, the OFB-DES algorithm uses the output of the previous DES cipher operation (not the exclusive-OR operation, such as CFB) in place of the initialization vector. (See **Reference 2** for information on how to decipher each mode.)

Attempts to break DES

Even a short search with an Internet search engine reveals that a lot of people are working to figure out how to break DES. In 1999, the EFF (Electronic Frontier Foundation) won the RSA DES Challenge III, in which competitors defeated DES encryption in 22 hours and 15 minutes, using a large network of PCs. The EFF also estimated that in that year, a $10 million machine could break DES in just 3.5 minutes.

Contributions in the areas of linear and differential cryptanalysis have lead to new estimates for DES cipher strength. These methods have decreased the DES cipher to 46 to 47 bits. For more information on these subjects, see **references 1** through **5**.

References

1. NIST DES standard document FIPS PUB 46-2, www.itl.nist.gov/fipspubs/fip46-2.htm.
2. NIST DES Modes of Operation FIPS PUB 81, www.itl.nist.gov/fipspubs/fip81.htm.
3. Handbook of Applied Cryptography, www.cacr.math.uwaterloo.ca/hac/.
4. Schneier, Bruce, *Applied Cryptography*, John Wiley and Sons, 1996.
5. Electronic Frontier Foundation, www.eff.org/descracker.html.

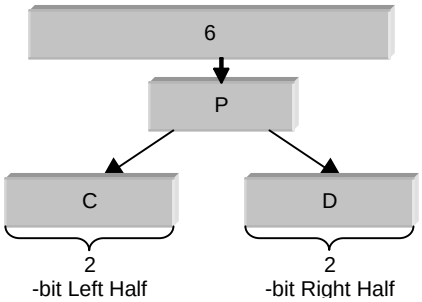

Figure 1-- Before the first round, the DES key derivation algorithm puts the 56-bit key through a key permutation. This operation both rearranges the bits in the key and splits the key into two 28-bit halves.

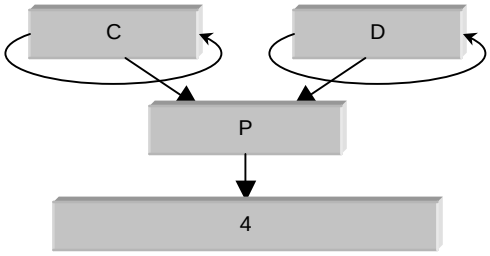

Figure 2-- When deriving the key for the first, second, ninth, and 16th round, the left and right halves rotate left through one bit position. For all other rounds, the halves rotate left through two bit positions.

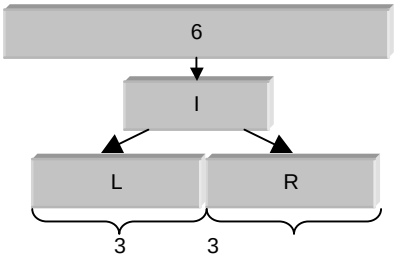

Figure 3--To be DES-compliant, the DES algorithm specifies that the 64-bits of plaintext must first go through an initial permutation.

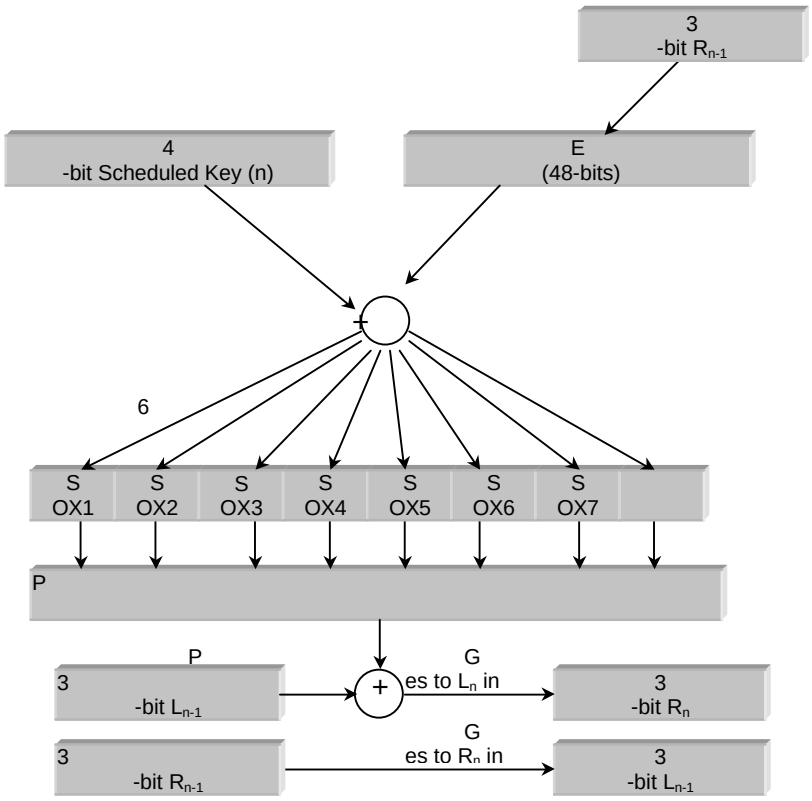

Figure 4-- The DES round is the core of the DES algorithm. Each 64-bit plaintext, post-initial permutation goes through the DES round 16 times during the encryption processing.

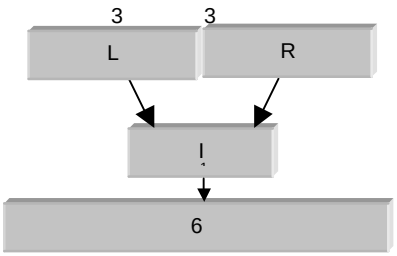

Figure 5--The final permutation moves bits from the left and right data halves into the final output.

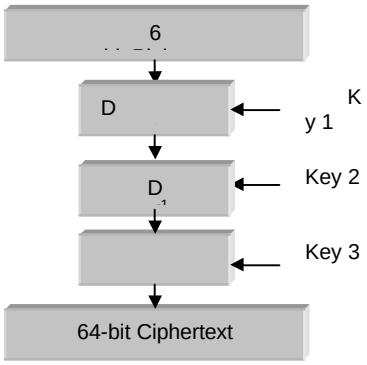

Figure 6--Triple-DES involves nothing more than running DES three times with three 56-bit keys.

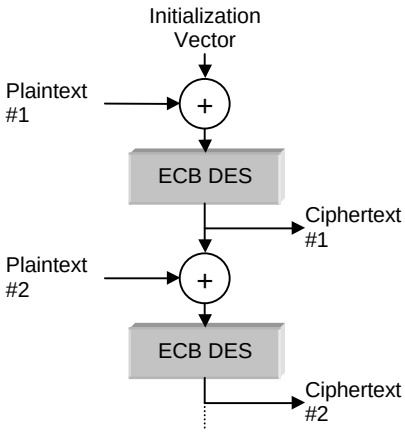

Figure 7-- The CBC (Cipher Block Chaining) mode is a method for protecting against fraudulent data insertion into the data stream.

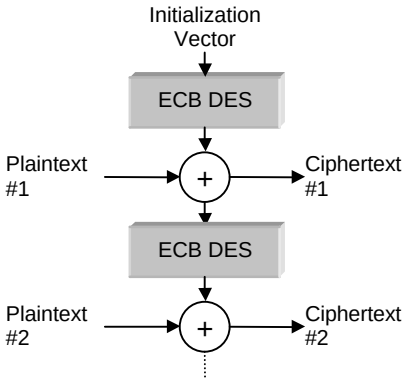

Figure 8—CFB (Cipher Feedback) is a more complicated mode of DES

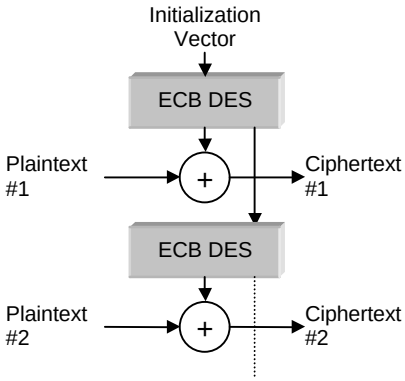

Figure 9—OFB (Output Feedback) is a slightly modified CFB (Cipher Feedback) mode.

Table 1--C_0 permutation (PC-1)

Permuted bit number (MSB)													
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
57	49	41	33	25	17	9	1	58	50	42	34	26	18
Key bit number													
15	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**25**	**27**	**28**
10	2	59	51	43	35	27	19	11	3	60	52	44	36

Table 2--D_0 permutation (PC-1)

Permuted bit number (MSB)													
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
63	55	47	39	31	23	15	7	62	54	46	38	30	22
Key bit number													
15	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**25**	**27**	**28**
14	6	61	53	45	37	29	21	13	5	28	20	12	4

Table 3--Key compression permutation (PC-2)

Permuted bit number (MSB)											
1	2	3	4	5	6	7	8	9	10	11	12
14	17	11	24	1	5	3	28	15	6	21	10
Combined halves bit number											
13	14	15	16	17	18	19	20	21	22	23	24
23	19	12	4	26	8	16	7	27	20	13	2
25	26	27	28	29	30	31	32	33	34	35	36
41	52	31	37	47	55	30	40	51	45	33	48
37	38	39	40	41	42	43	44	45	46	47	48
44	49	39	56	34	53	46	42	50	36	29	32

Table 4--Initial data permutation

Permuted bit number (MSB)											
1	2	3	4	5	6	7	8	9	10	11	12
58	50	42	34	26	18	10	2	60	52	44	36
Plain-text data bit											
13	14	15	16	17	18	19	20	21	22	23	24
28	20	12	4	62	54	46	38	30	22	14	6
25	26	27	28	29	30	31	32	33	34	35	36
64	56	48	40	32	24	16	8	57	49	41	33
37	38	39	40	41	42	43	44	45	46	47	48
25	17	9	1	59	51	43	35	27	19	11	3
49	50	51	52	53	54	55	56	57	58	59	60
61	53	45	37	29	21	13	5	63	55	47	39
61	62	63	64								
31	23	15	7								

Table 5--Expansion permutation (EP)

Permuted bit number (MSB)											
1	2	3	4	5	6	7	8	9	10	11	12
32	1	2	3	4	5	4	5	6	7	8	9
Right-half bit number											
13	14	15	16	17	18	19	20	21	22	23	24
8	9	10	11	12	13	12	13	14	15	16	17
25	26	27	28	29	30	31	32	33	34	35	36
16	17	18	19	20	21	20	21	22	23	24	25
37	38	39	40	41	42	43	44	45	46	47	48
24	25	26	27	28	29	28	29	30	31	32	1

Table 6--S-box 1 result

	S_1S_6			
$S_{2...5}$	0	1	2	3
0	14	0	4	15
1	4	15	1	12
2	13	7	14	8
3	1	4	8	2
4	2	14	13	4
5	15	2	6	9
6	11	13	2	1
7	8	1	11	7
8	3	10	15	5
9	10	6	12	11
10	6	12	9	3
11	12	11	7	14
12	5	9	3	10
13	9	5	10	0
14	0	3	5	6
15	7	8	0	13

Table 7--S-box 2 results

$S_{8...11}$	S_7S_{12} 0	1	2	3
0	15	3	0	13
1	1	13	14	8
2	8	4	7	10
3	14	7	11	1
4	6	15	10	3
5	11	2	4	15
6	3	8	13	4
7	4	14	1	2
8	9	12	5	11
9	7	0	8	6
10	2	1	12	7
11	13	10	6	12
12	12	6	9	0
13	0	9	3	5
14	5	11	2	14
15	10	5	15	9

Table 8--S-box 3 results

$S_{14...17}$	$S_{13}S_{18}$ 0	1	2	3
0	10	13	13	1
1	0	7	6	10
2	9	0	4	13
3	14	9	9	0
4	6	3	8	6
5	3	4	15	9
6	15	6	3	8
7	5	10	0	7
8	1	2	11	4
9	13	8	1	15
10	12	5	2	14
11	7	14	12	3
12	11	12	5	11
13	4	11	10	5
14	2	15	14	2
15	8	1	7	12

Table 9--S-box 4 results

$S_{20...23}$	$S_{19}S_{24}$ = 0	1	2	3
0	7	13	10	3
1	13	8	6	15
2	14	11	9	0
3	3	5	0	6
4	0	6	12	10
5	6	15	11	1
6	9	0	7	13
7	10	3	13	8
8	1	4	15	9
9	2	7	1	4
10	8	2	3	5
11	5	12	14	11
12	11	1	5	12
13	12	10	2	7
14	4	14	8	2
15	15	9	4	14

Table 10--S-box 5 results

$S_{26...29}$	$S_{25}S_{30}$ = 0	1	2	3
0	2	14	4	11
1	12	11	2	8
2	4	2	1	12
3	1	12	11	7
4	7	4	10	1
5	10	7	13	14
6	11	13	7	2
7	6	1	8	13
8	8	5	15	6
9	5	0	9	15
10	3	15	12	0
11	15	10	5	9
12	13	3	6	10
13	0	9	3	4
14	14	8	0	5
15	9	6	14	3

Table 11--S-box 6 results

$S_{32...35}$	$S_{31}S_{36}$ = 0	1	2	3
0	12	10	9	4
1	1	15	14	3
2	10	4	15	2
3	15	2	5	12
4	9	7	2	9
5	2	12	8	5
6	6	9	12	15
7	8	5	3	10
8	0	6	7	11
9	13	1	0	14
10	3	13	4	1
11	4	14	10	7
12	14	0	1	6
13	7	11	13	0
14	5	3	11	8
15	11	8	6	13

Table 12--S-box 7 results

$S_{38...41}$	$S_{37}S_{42}$ 0	1	2	3
0	4	13	1	6
1	11	0	4	11
2	2	11	11	13
3	14	7	13	8
4	15	4	12	1
5	0	9	3	4
6	8	1	7	10
7	13	10	14	7
8	3	14	10	9
9	12	3	15	5
10	9	5	6	0
11	7	12	8	15
12	5	2	0	14
13	10	15	5	2
14	6	8	9	3
15	1	6	2	12

Table 13--S-box 8 results

$S_{44...47}$	$S_{43}S_{48}$ 0	1	2	3
0	13	1	7	2
1	2	15	11	1
2	8	13	4	14
3	4	8	1	7
4	6	10	9	4
5	15	3	12	10
6	11	7	14	8
7	1	4	2	13
8	10	12	0	15
9	9	5	6	12
10	3	6	10	9
11	14	11	13	0
12	5	0	15	3
13	0	14	3	5
14	12	9	5	6
15	7	2	8	11

Table 14--P permutation (PP)

Permuted bit number (MSB)											
1	2	3	4	5	6	7	8	9	10	11	12
16	7	20	21	29	12	28	17	1	15	23	26
S-box bit number											
13	14	15	16	17	18	19	20	21	22	23	24
5	18	31	10	2	8	24	14	32	27	3	9
25	26	27	28	29	30	31	32				
19	13	30	6	22	11	4	25				

Table 15--Final permutation

Cipher text permuted bit number (MSB)											
1	2	3	4	5	6	7	8	9	10	11	12
40	8	48	16	56	24	64	32	39	7	47	15
Left/right-half data bit											
13	14	15	16	17	18	19	20	21	22	23	24
55	23	63	31	38	6	46	14	54	22	62	30
25	26	27	28	29	30	31	32	33	34	35	36
37	5	45	13	53	21	61	29	36	4	44	12
37	38	39	40	41	42	43	44	45	46	47	48
52	20	60	28	35	3	43	11	51	19	59	27
49	50	51	52	53	54	55	56	57	58	59	60
34	2	42	10	50	18	58	26	33	1	41	9
61	62	63	64								
49	17	57	25								