Block Ciphers

Block Ciphers and Stream Ciphers

In practical ciphers the plaintext M is divided into fixed-length blocks $M = M_1 M_2 \ldots M_N$. Then, each block M_i is encrypted to the ciphertext block $C_i = E_K(M_i)$, and the results are concatenated to the ciphertext $C = C_1 C_2 \ldots C_N$.

There are two major kind of ciphers, which differ in the way the plaintexts are encrypted:

Stream Ciphers

The blocks are encrypted sequentially, each block is encrypted by a distinct transformation, which might depend on

1. the previous encrypted blocks,
2. the previous transformation,
3. the block number,
4. the key.

This information from one block is kept in **memory** between the encryption of this block and the succeeding block, for use during the encryption of the succeeding block.
Usually, stream ciphers use **blocks of either one bit or eight bits** (one character).

Block Ciphers

All the blocks are encrypted in the same way, under exactly the same transformation (no memory): $C_1 = E(M_1)$, $C_2 = E(M_2)$, etc.

Encryption transformation should not be vulnerable to known plaintext attacks. Attacker should not be able to collect (almost) all the plaintext/ciphertext blocks pairs, keep the transformation table $T(M) = C$, and use it to encrypt/decrypt if they do not know the mathematical formulation of the transformation (and in particular the key).

Thus, **the block size should be large**, and the number of distinct possible values in a plaintext block should be larger than the minimal allowed complexity of an attack.

In the past **blocks of 64 bits** were used, which have 2^{64} possibilities, whose table storing costs at least 2^{64} known plaintexts and memory space.

Nowadays, the standard block size is 128 bits.

Block Ciphers

Block ciphers are substitution ciphers in which the plaintext and the ciphertext blocks are binary vectors of length N. When $N = 64$ there are 2^{64} different plaintexts/ciphertexts, and when $N = 128$ there are 2^{128} different plaintexts/ciphertexts.

For each key the encryption function $E_K(\cdot)$ is a permutation from $\{0,1\}^N$ to itself.

$D_K(\cdot)$ is the decryption function (the inverse permutation),
such that $D_K(E_K(\cdot)) = E_K(D_K(\cdot)) = \text{Identity}$.

The Data Encryption Standard - DES

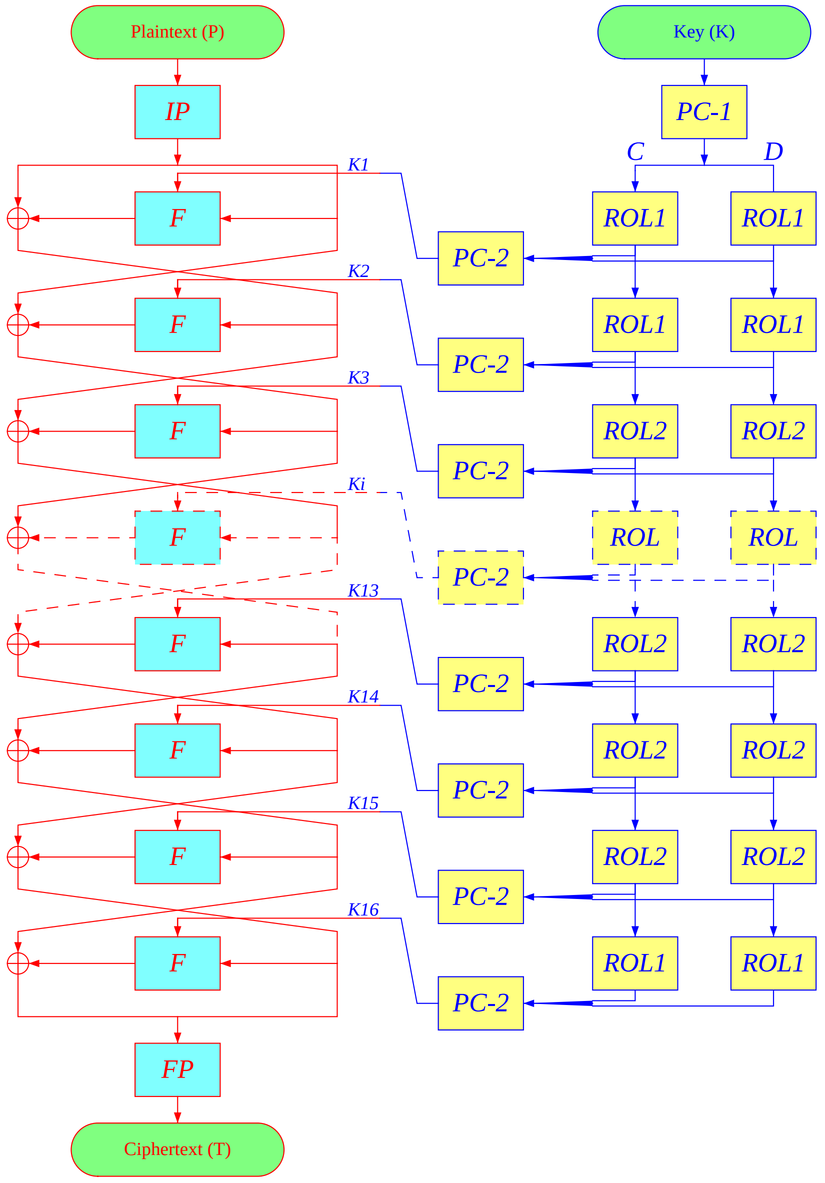
1. The most widely used cipher in civilian applications.
2. Developed by IBM; Evolved from Lucifer.
3. Accepted as an US NBS standard in 1977, and later as an international standard.
4. A block cipher with ***N* = 64 bit blocks**.
5. **56-bit keys** (eight bytes, in each byte seven bits are used; the eighth bit can be used as a parity bit).
6. Exhaustive search requires 2^{56} encryption steps (2^{55} on average).

The Data Encryption Standard - DES (cont.)

7. Iterates a round-function 16 times in 16 **rounds**. The round-function mixes the data with the key.

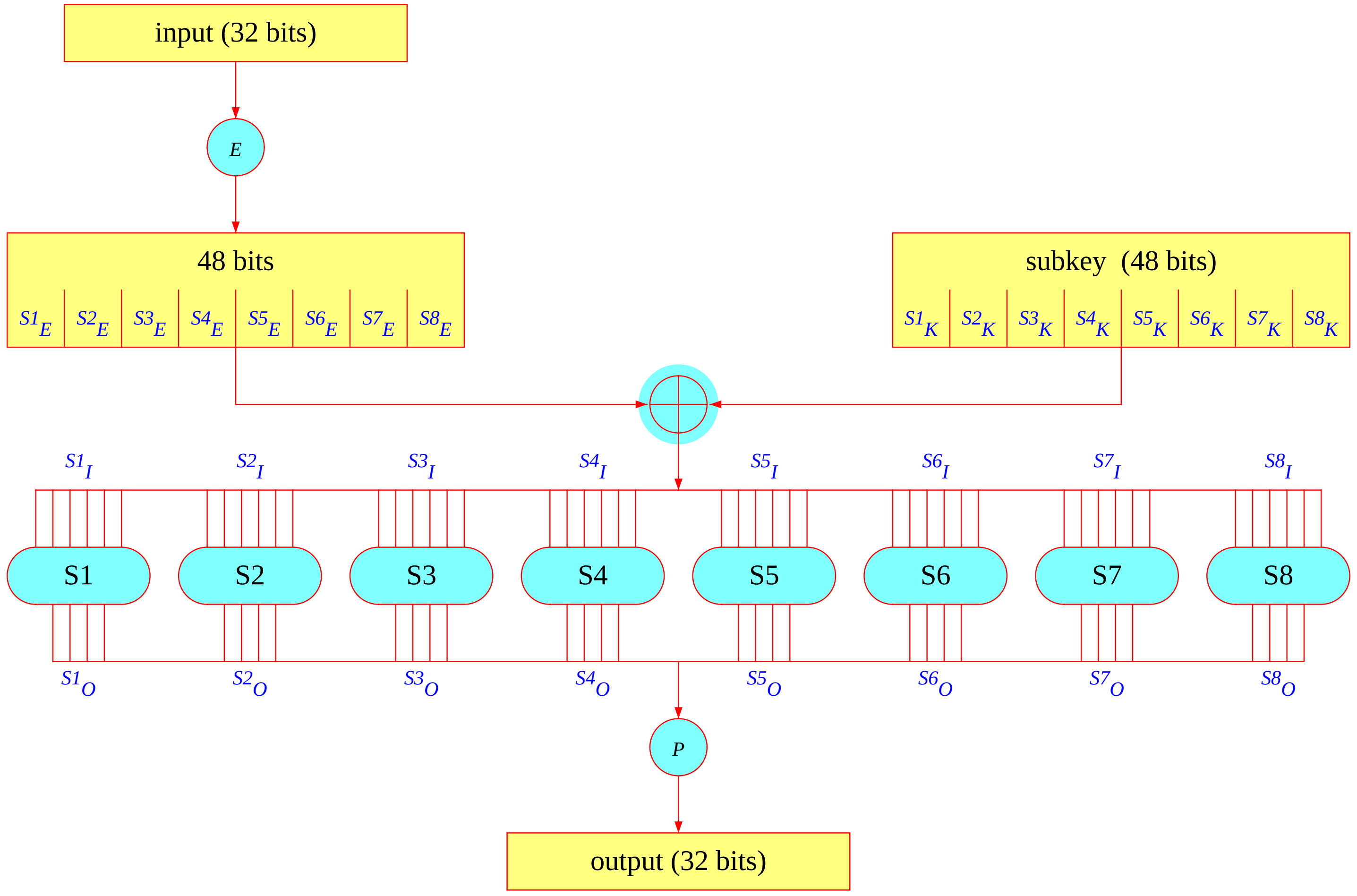
8. Each round, the key information entered to the round function is called a **subkey**. The subkeys $K_1, \ldots, K_{16}$ are computed by a **key scheduling algorithm**.

DES Outline



Plaintext (P)
IP
F
K1
K2
K3
Ki
K13
K14
K15
K16
FP
Ciphertext (T)
Key (K)
PC-1
C
D
ROL1
ROL2
ROL
PC-2

The F-Function


input (32 bits)
E
48 bits
S1E S2E S3E S4E S5E S6E S7E S8E
subkey (48 bits)
S1K S2K S3K S4K S5K S6K S7K S8K
S1I S2I S3I S4I S5I S6I S7I S8I
S1 S2 S3 S4 S5 S6 S7 S8
S1O S2O S3O S4O S5O S6O S7O S8O
P
output (32 bits)

The Initial Permutation (IP)

The following tables describe for each output bit the number of the input bit whose value enters to the output bit. For example, in IP, the 58'th bit in the input becomes the first bit of the output.

IP:

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

FP=IP^{-1}:

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

The P Permutation and the E Expansion

P Permutes the order of 32 bits. E Expands 32 bits to 48 bits by duplicating 16 bits twice.

P:

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

E:

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

The S Boxes

S box S1:

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S box S2:

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

The S Boxes (cont.)

S box S3:

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S box S4:

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

The S Boxes (cont.)

S box S5:

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S box S6:

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

The S Boxes (cont.)

S box S7:

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S box S8:

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

The S Boxes (cont.)

How to interpret the S boxes:
The representation of the S boxes use the first and sixth bits of the input as a line index (between 0 and 3), and the four middle bits as the row index (between 0 and 15).
Thus, the input values which correspond to the standard description of the S boxes are

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31
32	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62
33	35	37	39	41	43	45	47	49	51	53	55	57	59	61	63

The S Boxes (cont.)

Note that **all the operations are linear, except for the S boxes**. Thus, **the strength of DES crucially depends on the choice of the S boxes**.

If the S boxes would be affine, the cipher becomes affine, and thus easily breakable.

The S boxes were chosen with some criteria to prevent attacks.

The Key Scheduling Algorithm

The key scheduling algorithm generates the 16 48-bit subkeys from the 56-bit key, by duplicating each key bit into about 14 of the subkeys in a particular order.

PC-1:

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Number of rotations in the key scheduling algorithm:

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Rotations	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

The Key Scheduling Algorithm (cont.)

PC-2:

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Decryption

Decryption is done by the **same algorithm** as encryption, except that the order of the subkeys is reversed (i.e., K16 is used instead of K1, K15 instead of K2, . . . , and K1 instead of K16.).

Known Weaknesses

1. **Secret design criteria**: IBM were requested by the NSA not to publish the design criteria (later known as differential cryptanalysis).
2. **Weak keys**: four keys for which $E_K(E_K(X)) = X$ for any X (i.e., $E_K(\cdot) \equiv D_K(\cdot)$).
3. **Semi-weak keys**: 12 keys (6 pairs) for which $E_{K_1}(E_{K_2}(X)) = X$ for any X (i.e., $E_{K_1}(\cdot) \equiv D_{K_2}(\cdot)$).
4. **The S boxes are not random**: for example,
   (a) S4 is 75% redundant: Each line of S4 can be calculated from any other line by a linear function.
   (b) In all the S boxes ($S(x_1x_2x_3x_4x_5x_6) = y_1y_2y_3y_4$) there is a high correlation between the bit x_2 of the input to the parity of the output $y_1 \oplus y_2 \oplus y_3 \oplus y_4$ (For example, in S5 there are 12 equalities and 52 inequalities).

Known Weaknesses (cont.)

5. **Short 56-bit keys**: On a single processor exhaustive search takes:

   (a) In the 70's: thousands of years.

   (b) In the 90's: On DEC GaAs DES chip: 100 years.

   Exhaustive search is not totally infeasible:

   (a) In the 70's: Diffie and Hellman suggested a 20,000,000$ special purpose machine which can find a key within a day. They estimate the cost per solution to be 5000$.

   (b) In the 90's: Wiener suggested a 1,000,000$ special purpose machine which can find a key within 3.5 hours.

   (c) June 1997: Done over the *Internet* using 14000–78000 computers within 90 days.

   (d) January 1998: Done over the Internet within 39 days.

   (e) July 1998: A $210000 search machine, called *deep crack*, was built. Found a key within 56 hours.

Known Weaknesses (cont.)

(f) October 2006: COPACOBANA, a machine that costs €9000 and finds a DES key in 17 days (worst case) is presented.

(g) March 2007: Newer COPACOBANA, similar price, and about 13 days for finding a DES key.

(h) ...

The Complementation Property

Let $\bar{X}$ be the 1-complement of X.
In DES if

$$C = E_K(P)$$

then

$$\bar{C} = E_{\bar{K}}(\bar{P}).$$

Exercise: Prove the complementation property.

The Complementation Property (cont.)

This property can be used for a chosen plaintext attack, twice faster than exhaustive search:

1. Choose P.
2. Ask for the ciphertexts $C_0 = E_K(P)$ and $C_1 = E_K(\bar{P})$.
3. Exhaustively try all the 2^{55} keys K' whose most significant bit is zero, whether $E_{K'}(P) \in \{C_0, \bar{C}_1\}$.
4. if $E_{K'}(P) = C_0$, then probably $K = K'$.
5. if $E_{K'}(P) = \bar{C}_1$, then probably $K = \bar{K}'$.
6. Otherwise, neither K' nor $\bar{K}'$ are the key K.
7. Since comparisons are much faster than a trial encryptions, this attack is twice faster than exhaustive search.

Contemporary Attacks on DES

Today there are three methods which can reduce the complexity of attacks to less than 2^{55}. All of them require a huge (unrealistic) amount of data (their complexity is the time to create the data, while the analysis is much faster).

Differential cryptanalysis: 2^{47} chosen plaintexts (1990–1991, Biham and Shamir).

Linear cryptanalysis: 2^{43} known plaintexts (1993–1994, Matsui). Slightly improved to $0.75 \cdot 2^{43}$ (1998, Shimoyama, Kaneko).

Improved Davies attack: 2^{50} known plaintexts (1994, Biham and Biryukov).

Statistical Cryptanalysis: $2^{42.9}$ known plaintexts (1996, Vaudenay).

Contemporary Attacks on DES (cont.)

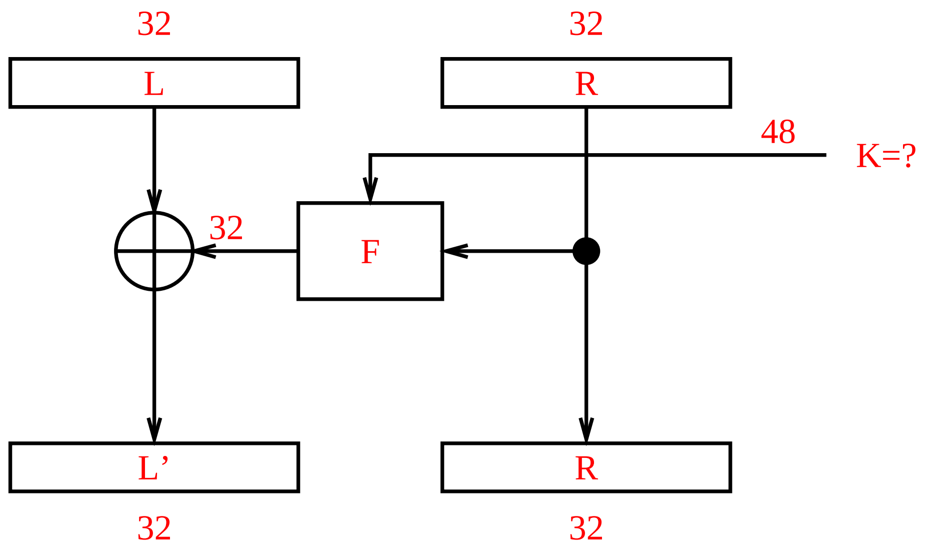
Related Keys: (1993, Biham) proposes that the shifts in the key scheduling algorithm should not be all equal. If they were all equal (like all 2, or all 7) then DES could be broken with either

- $1.43 \cdot 2^{53}$ steps, or
- 2^{33} chosen key known plaintexts, or
- 2^{17} chosen key chosen plaintexts.

Key-Collision Attacks: (Biham, 1996) In some circumstances it is much easier to "forge" messages than to "break" the cipher: 2^{28}.

A Known Plaintext Attack on 1-Round DES

After removing the permutations IP and FP we get:



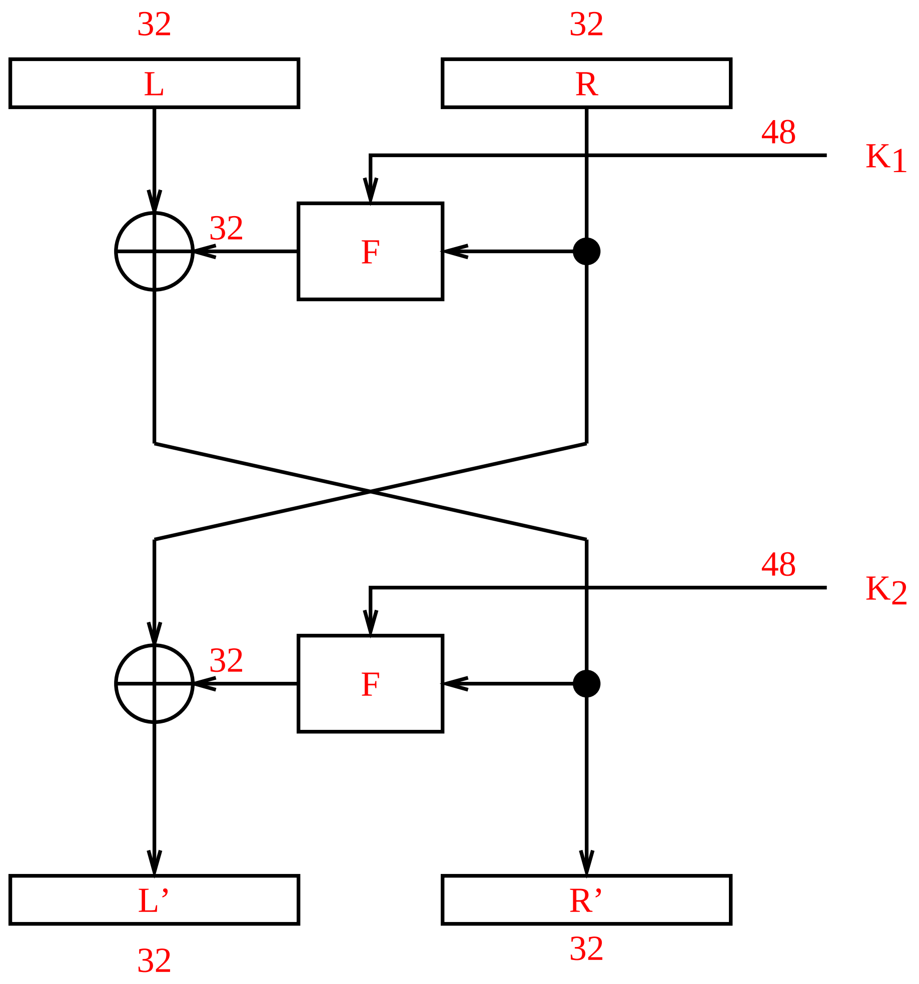
A Known Plaintext Attack on 1-Round DES (cont.)

We are given a pair (M, C) where $M = (L, R)$ and $C = (L', R)$ and we want to find the 48-bit key K.
We know that:

$$F(R, K) = L \oplus L'$$

1. Why is the output of all S-boxes known?
2. Given the 4 bits output of S_1 how many 6-bit combinations are possible as input to S_1?
3. How many 6-bit combinations are possible as the 6 bit key which takes part in the creation of the input to S_1?
4. How many 48-bit combinations are possible for K?

A Known Plaintext Attack on 2-Round DES



A Known Plaintext Attack on 2-Round DES (cont.)

Thus, we have:

- $F(R, K_1) = L \oplus R'$
- $F(R', K_2) = L' \oplus R$

As in the attack on one round, the first expression reduces the number of possibilities for the 48 bits of K_1 to $4^8 = 2^{16}$ (as only a fraction of 2^{-32} of the keys pass the test).
The second expression reduces the number of possibilities for the 48 bits of K_2 to $4^8 = 2^{16}$ as well.
The set of possibilities for K_1 and K_2 intersects on the 40 bits, which are common to both round keys K_1, K_2. As the right key must appear, we get that the average number of possible 56-bit keys is slightly above 1.

A Known Plaintext Attack on 4-Round DES

After 4 rounds there are still bits which are unaffected by some key bits. For example:

- Original key bit 46 is not used in round 1.
- It is used in round 2 (in K_2) for the creation of the input to S_5, thus it affects the 4 bits at the output of S_5.
- These 4 bits become bits $8, 14, 25$ and 3 after the permutation P.
- Key bit 46 is not used in round 3.
- Bits $8, 14, 25, 3$ affect the output of $S_1, S_2, S_3, S_4, S_6, S_7$ in round 3, thus leaving the output bits of S_5 and S_8 unaffected by key bit 46.
- There are 8 bits in the left 32-bit output of round 3 unaffected from key bit 46.
- There are 8 bits in the right 32-bit output of round 4. unaffected from key bit 46 (but they are affected by the other key bits and by the plaintext).

A Known Plaintext Attack on 4-Round DES (cont.)

This property can be used for a known plaintext attack:

1. Fix key bit 46 to be zero.
2. For every key K^1 with the key bit 46 set to zero (there are 2^{55} such keys):
   (a) Encrypt the plaintext: $C' = E_{K^1}(P)$.
   (b) Compare the 8 bits which are unaffected by key bit 46 in C' to the same 8 bits in the given ciphertext C.
   (c) If those bits are the same:
   Denote K^1 with the 46'th bit set to one by K^2.
   If $E_{K^1}(P) = C$ then K^1 is probably the key.
   If $E_{K^2}(P) = C$ then K^2 is probably the key.
   If neither, continue to next key.

A Known Plaintext Attack on 4-Round DES (cont.)

Analysis: We encrypt the plaintext with all 2^{55} possibilities for the key K^1. 2^{47} keys on average agree with the ciphertext on the 8 bits. Therefore, we encrypt 2^{47} possible K^2 values in order to find the original key. The resulting time complexity is $2^{55} + 2^{47}$ encryptions instead of 2^{56}.
Remark: When using the complementation property with this attack we get $2^{54} + 2^{46}$ encryptions in the worst case, and $2^{53} + 2^{45}$ encryptions in the average case.

A Known Plaintext Attack on 5-Round DES

Key bit 52 can be also used for an attack on 4 rounds:

- Original key bit 52 is not used in round 1.
- It is used in round 2 (in K_2) for the creation of the input to S_1, thus it affects the 4 bits at the output of S_1.
- These 4 bits become bits $9, 17, 23$ and 31 after the permutation P.
- Key bit 52 is not used in round 3.
- Bits $9, 17, 23, 31$ affect the output of S_2, S_3, S_4, S_5, S_6, S_8 in round 3, thus leaving the output bits of S_1 and S_7 unaffected by key bit 52.
- There are 8 bits in the left 32 bit output of round 3 unaffected from key bit 52.
- There are 8 bits in the right 32 bit output of round 4 unaffected from key bit 52.

Key bit 52 has another property — it is not used in the 5'th round.

A Known Plaintext Attack on 5-Round DES (cont.)

Thus, we can use it for the following known plaintext attack:

1. Fix key bit 52 to be zero.
2. For every key K^1 with the key bit 52 set to zero (there are 2^{55} such keys):
   (a) Encrypt the plaintext up to round 4: $E^4_{K^1}(P)$.
   (b) Decrypt the ciphertext one round: $D^1_{K^1}(C)$.
   (c) Compare the 8 bits which are unaffected by key bit 52 in $E^4_{K^1}(P)$ to the same 8 bits in $D^1_{K^1}(C)$.
   (d) If those bits are the same:
   Denote K^1 with the 52'th bit set to one by K^2.
   If $E_{K^1}(P) = C$ then K^1 is probably the key.
   If $E_{K^2}(P) = C$ then K^2 is probably the key.

A Known Plaintext Attack on 5-Round DES (cont.)

Analysis: We encrypt the plaintext with all 2^{55} possibilities for the key K^1 with key bit 52 set to zero. We are left with 2^{47} keys on average. We also encrypt these 2^{47} possible K^2 in order to find the original key. Thus, we perform $2^{55}+2^{47}$ encryptions instead of 2^{56}.
Remark: As in the attack on 4 rounds when using the complementation property we get $2^{54}+2^{46}$ encryptions in the worst case, and $2^{53}+2^{45}$ encryptions in the average case.

Modes of Operation

Long messages of several blocks are encrypted using one of the **modes of operation**. In the modes of operation, the messages M are divided into N-bit blocks $M_1M_2\ldots M_n$, each block M_i is encrypted (as defined by the mode of operation) to C_i, and the results are concatenated into the ciphertext $C=C_1C_2\ldots C_n$.

Many modes of operation actually transform the block ciphers into stream ciphers, by adding memory (external to the block cipher).

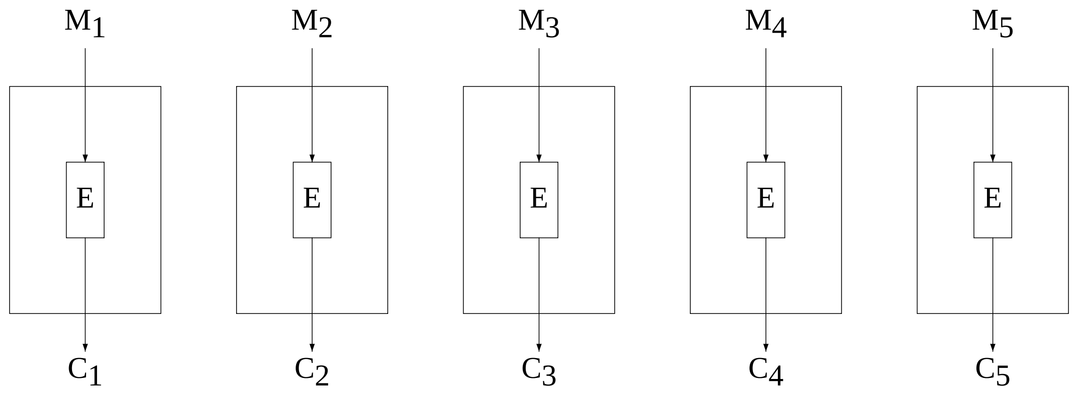
Electronic Code Book (ECB) Mode

In this mode, each plaintext block M_i is encrypted simply by

$$C_i = E_K(M_i).$$

Decryption is done by

$$M_i = D_K(C_i).$$



The main drawback of ECB is that $M_i = M_j$ iff $C_i = C_j$. Thus, Eve can easily identify which plaintext blocks are equal.

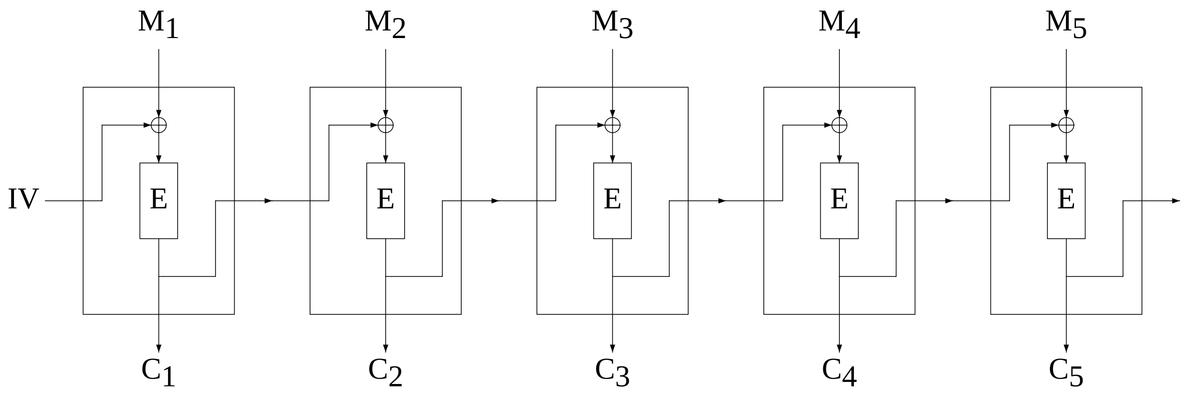
Cipher Block Chaining (CBC) Mode

Each plaintext block M_i is mixed with the previous ciphertext block before encryption:

$$C_i = E_K(M_i \oplus C_{i-1}).$$

Decryption is done by

$$M_i = D_K(C_i) \oplus C_{i-1}.$$



A (non-secret) initial value $C_0 = IV$ is chosen for each message.
In this mode two equal message blocks are usually encrypted to different ciphertext blocks.

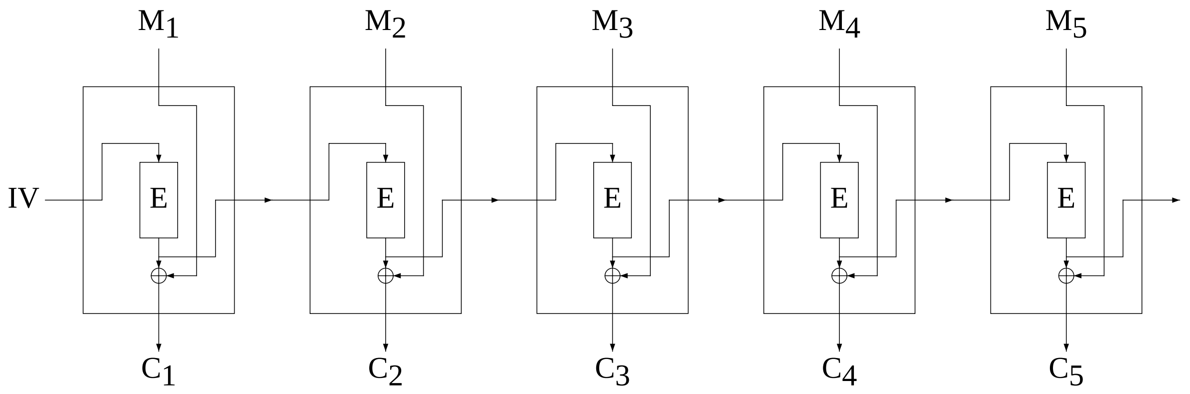
Cipher Block Chaining (CBC) Mode (cont.)

This mode has small error propagation: if C_i is received with errors, only M_i and M_{i+1} have errors after decryption. M_{i+2} is not affected from the error in C_i.

Output FeedBack (OFB) Mode

Generates a pseudo random bit stream from the key K and an initial value $V_0 = IV$ by

$$V_i = E_K(v_{i-1}).$$



Encryption is done by

$$C_i = M_i \oplus V_i.$$

Decryption is done by

$$M_i = C_i \oplus V_i.$$

This mode has no error propagation (one bit error in C causes only one bit error in the message during decryption).

Output FeedBack (OFB) Mode (cont.)

An advantage of this mode is that V_i can be computed in advance, before the plaintext/ciphertext is known.

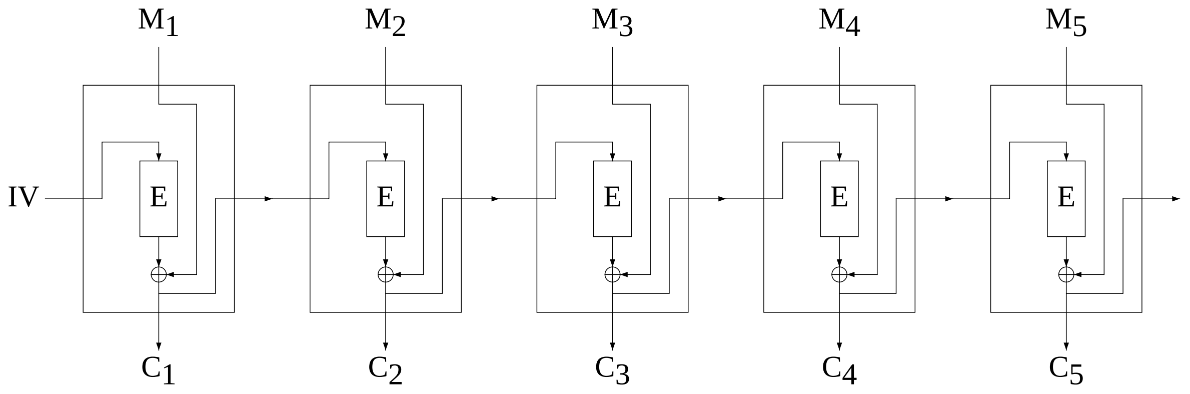
Cipher FeedBack (CFB) Mode

Similar to the OFB mode, but the bit stream depends on the ciphertext. Encryption is done by

$$C_i = M_i \oplus E_K(C_{i-1}).$$

Decryption is done by

$$M_i = C_i \oplus E_K(C_{i-1}).$$



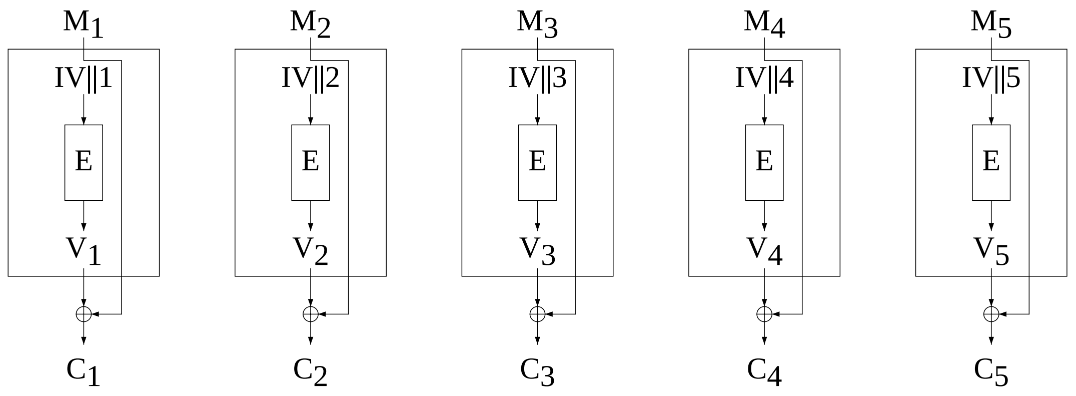
Counter Mode (CTR) Mode

Counter mode accepts a nonce (an IV which is used only once with a given key), and operates encryption in a very similar manner to OFB, where the input to the block cipher in the ith iteration is $IV||i$. Encryption is done by

$$C_i = M_i \oplus E_K(IV||i).$$

Decryption is done by

$$M_i = C_i \oplus E_K(IV||i).$$



Counter Mode (CTR) Mode (cont.)

Note that $V_i = E_k(IV||i)$ can be computed in advance just like in OFB, but can also be generated "on demand" (i.e., computing V_3 does not require the knowledge of V_2).

On the other hand, just like in OFB, the same (key, IV) combination must not be repeated.

Triple-DES

Triple-DES is an improvement to DES which use the same DES hardware/ software but encrypt the plaintext three times.
Triple DES has two variants:

1. **Three-key triple-DES**: the plaintext is encrypted three times under three different DES keys K_1, K_2, and K_3.

   $$C = \mathrm{DES}_{K_3}(\mathrm{DES}^{-1}_{K_2}(\mathrm{DES}_{K_1}(P))).$$

   The total key length is $3 \cdot 56 = 168$ bits.

2. **Two-key triple-DES**: the plaintext is encrypted three times under two different DES keys K_1, and K_2, where K_1 is used in the first and third application of DES, and K_2 in the second application.

   $$C = \mathrm{DES}_{K_1}(\mathrm{DES}^{-1}_{K_2}(\mathrm{DES}_{K_1}(P))).$$

   The total key length is $2 \cdot 56 = 112$ bits.

Triple-DES (cont.)

Note that the second application of DES performs decryption. This is done to allow compatibility to older systems which use DES: if the two or three keys are all equal, then the triple encryption actually performs a single encryption with that key.

Speed:

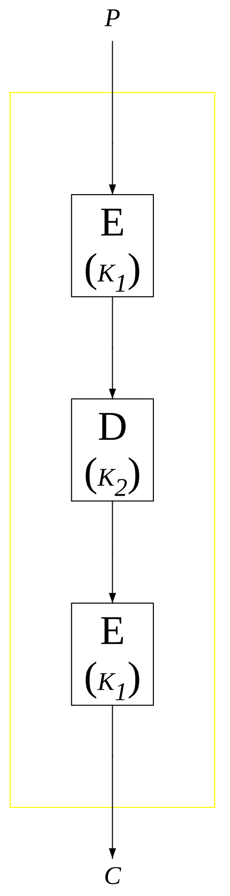
Triple-DES is about three times slower than DES. (It is slightly faster than a third of the speed of DES as the initial and final permutation in the borders between the first/second and the second/third DES applications can be eliminated).

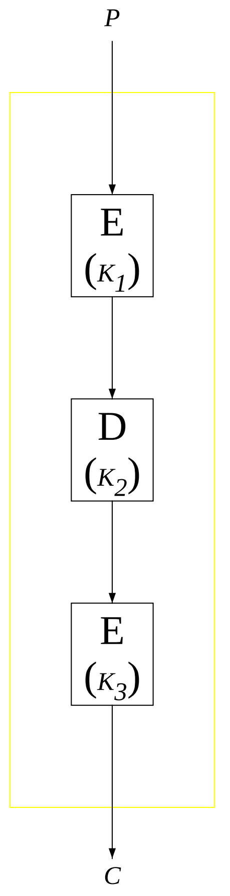
Status:

Triple-DES replaced DES as the de-facto standard a few years ago. AES is now the standard for new applications, while Triple-DES remains in many old applications.

Triple-DES (cont.)

Two-key and three-key Triple-DES:





Strength of Triple-DES

Triple-DES is believed to be secure for any practical purpose.

Its key size of 112 and 168 bits ensures that brute force (exhaustive search) attacks are impossible.

Other attacks are eliminated due to the large number of rounds.

The best theoretical known attack on Three-key Triple-DES has complexity 2^{108}.

Two-key Triple-DES has a theoretical attack with complexity 2^{56} (2^{56} time and 2^{56} chosen plaintext blocks). Although this attack is impractical, it is preferable to use Three-key Triple-DES.

Skipjack and the Clipper Chip

In 1993 the US government started an initiative to give the industry an NSA-developed cipher, called **Skipjack**, with a military strength (used for non-secret information). The catch is that the design of the cipher will not be published, and that the users of the system will have to disclose their keys to the government agencies if the court decides to. To ensure that, the cryptographic chip, called the **Clipper chip**, will send the key encrypted under a secret key which can be recovered if several government agencies collaborate (but not without collaboration).

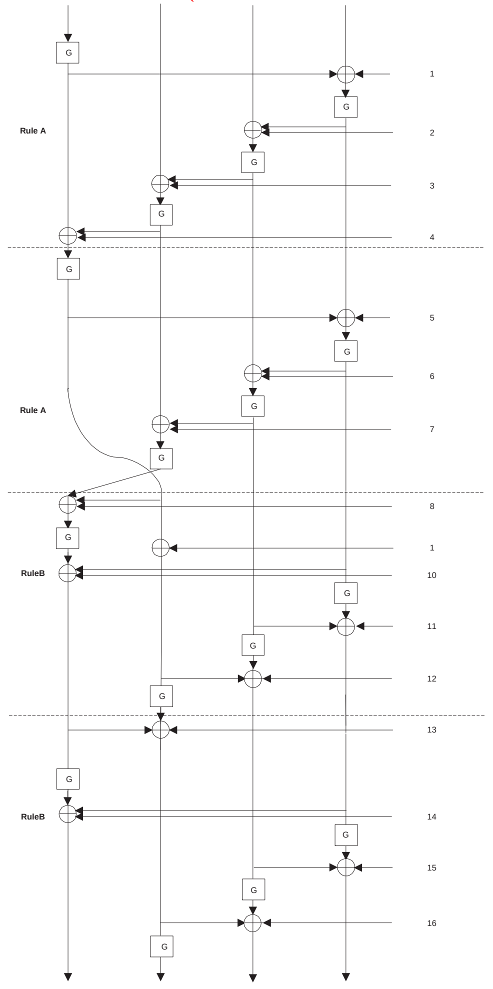
This initiative started a long debate, and did not survive.

Skipjack encrypts blocks of 64 bits, and have 80-bit keys.

In June 1998, Skipjack was published by NIST in order to allow implementation in software. Consecutively, it was analyzed, and shown that a reduced cipher with 31 rounds, rather than the 32 rounds of Skipjack, is theoretically breakable faster than exhaustive search.

Skipjack and the Clipper Chip (cont.)

The first half of Skipjack: (the second half is similar)



The Advanced Encryption Standard (AES)

AES is the new standard replacement for DES. It has blocks of 128 bits. It supports key sizes of 128, 192, and 256 bits.

In 1997 NIST (the institute which standardized DES) called for proposals for a successor for DES. 15 proposals were submitted from all over the world, and in August 1999 the list was reduced to 5 finalists:

- Mars (IBM; USA)
- RC6 (RSA data security; USA)
- Rijndael (Daemen, Rijmen; Belgium)
- Serpent (Anderson, Biham, Knudsen; UK, Israel, Norway)
- Twofish (Counterpane; USA)

In October 2000 **Rijndael** was selected to be the AES, and in November 2001 it became a formal standard (FIPS 197).

The AES – Rijndael

1. A block cipher with ***N* = 128 bit blocks**.
2. Supports three key sizes: **128** bits, **192** bits, and **256** bits
3. Exhaustive search requires 2^{128} (2^{192}, 2^{256}, respectively) encryption steps (2^{127}, 2^{191}, 2^{255} on average).
4. Iterates a round-function 10–14 times in 10–14 **rounds**. The round-function mixes the data with the key.
5. Each round, the key information entered to the round function is called a **subkey**. The subkeys $K_0, \ldots, K_{10}$ (or $K_0, \ldots, K_{14}$) are computed by a **key scheduling algorithm**.

Rijndael

Let the 128-bit blocks be divided into 32-bit words, and each word be divided into bytes.

Let the block be viewed in a rectangle (square) of 4×4 bytes, where each column have the bytes of a word

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

128-bit keys are viewed in the same way. Longer keys have additional columns, e.g., 256-bit keys

0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

Rijndael (cont.)

Rijndael uses four invertible operations:

1. SubBytes (S box; 8-bit to 8-bit)
2. ShiftRows (rotating order of bytes in each row)
3. MixColumns (linear mixing of a word column)
4. AddRoundKey (key mixing)

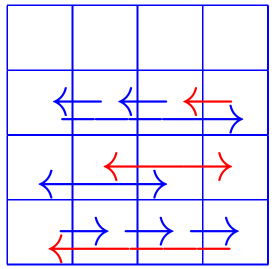
SubBytes

The byte substitution is a fixed S box from 8 bits to 8 bits. It substitutes all the bytes of the input by the following table

```
word8 S[256] = {
 99, 124, 119, 123, 242, 107, 111, 197,  48,   1, 103,  43, 254, 215, 171, 118,
202, 130, 201, 125, 250,  89,  71, 240, 173, 212, 162, 175, 156, 164, 114, 192,
183, 253, 147,  38,  54,  63, 247, 204,  52, 165, 229, 241, 113, 216,  49,  21,
  4, 199,  35, 195,  24, 150,   5, 154,   7,  18, 128, 226, 235,  39, 178, 117,
  9, 131,  44,  26,  27, 110,  90, 160,  82,  59, 214, 179,  41, 227,  47, 132,
 83, 209,   0, 237,  32, 252, 177,  91, 106, 203, 190,  57,  74,  76,  88, 207,
208, 239, 170, 251,  67,  77,  51, 133,  69, 249,   2, 127,  80,  60, 159, 168,
 81, 163,  64, 143, 146, 157,  56, 245, 188, 182, 218,  33,  16, 255, 243, 210,
205,  12,  19, 236,  95, 151,  68,  23, 196, 167, 126,  61, 100,  93,  25, 115,
 96, 129,  79, 220,  34,  42, 144, 136,  70, 238, 184,  20, 222,  94,  11, 219,
224,  50,  58,  10,  73,   6,  36,  92, 194, 211, 172,  98, 145, 149, 228, 121,
231, 200,  55, 109, 141, 213,  78, 169, 108,  86, 244, 234, 101, 122, 174,   8,
186, 120,  37,  46,  28, 166, 180, 198, 232, 221, 116,  31,  75, 189, 139, 138,
112,  62, 181, 102,  72,   3, 246,  14,  97,  53,  87, 185, 134, 193,  29, 158,
225, 248, 152,  17, 105, 217, 142, 148, 155,  30, 135, 233, 206,  85,  40, 223,
140, 161, 137,  13, 191, 230,  66, 104,  65, 153,  45,  15, 176,  84, 187,  22
};
```

ShiftRows

The ShiftRows operation rotates the order of bytes in the i'th row of the rectangle by i bytes to the left



MixColumns

MixColumns is a linear mixing of a word column (four bytes) (a_0, a_1, a_2, a_3) into a word column (b_0, b_1, b_2, b_3). The mixing is defined by operations in $GF(2^8)$.

$$\begin{bmatrix} s'_0 \\ s'_1 \\ s'_2 \\ s'_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

The field $GF(2^8)$ is constructed over the (irreducible) polynomial `11B`, i.e., $x^8 + x^4 + x^3 + x + 1$.

MixColumns (cont.)

A different representation is

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} f(a_0, a_1, a_2, a_3) \\ f(a_1, a_2, a_3, a_0) \\ f(a_2, a_3, a_0, a_1) \\ f(a_3, a_0, a_1, a_2) \end{pmatrix}$$

where

$$f(a_0, a_1, a_2, a_3) = (a_1 \oplus a_2 \oplus a_3) \oplus g((a_0 \oplus a_1) \ll 1)$$

$$g(x) = x\&100_x \,?\, x \oplus 11B_x : x$$

($\ll$, $\gg$, &, ? : are shifts, logical AND, and conditional expression as in the C programming language).

Note that the left shift might turn the 9th bit on. The XOR with $11B_x$ in $g()$ ensures that the result always fits in a byte ($g()$ enforces the Galois field GF(2^8) modulo the polynomial denoted by $11B_x$, i.e., $x^8 + x^4 + x^3 + x + 1$).

MixColumns (cont.)

The **inverse** of the MixColumns operation is also linear, and is similar to the form of MixColumns itself, where f is replaced by d where

$$\begin{aligned}
d_1(a_0, a_1, a_2, a_3) &= a_0 \oplus a_1 \oplus a_2 \oplus a_3 \\
d_2(a_0, a_1, a_2, a_3) &= g(d_1(a_0, a_1, a_2, a_3) \ll 1) \oplus (a_0 \oplus a_2) \\
d_3(a_0, a_1, a_2, a_3) &= g(d_2(a_0, a_1, a_2, a_3) \ll 1) \oplus (a_0 \oplus a_1) \\
d(a_0, a_1, a_2, a_3) &= g(d_3(a_0, a_1, a_2, a_3) \ll 1) \oplus (a_1 \oplus a_2 \oplus a_3)
\end{aligned}$$

AddRoundKey (key mixing)

The key mixing operation XORs the data with a subkey. The subkey has the same size of the blocks.

The subkeys are computed from the key by a key scheduling algorithm, shown later.

Encryption

Encryption is performed by

1. First key mixing (using subkey K_0)
2. Nr rounds consisting of
   (a) SubBytes
   (b) ShiftRows
   (c) MixColumns (except for the last round)
   (d) AddRoundKey (using subkey K_i in round $i \in \{1, \ldots, Nr\}$)

Decryption

Decryption applies the inverses of the operations in the reverse order, using the reverse order of the subkeys.
It requires the application of the invSubBytes, invMixColumns, and invShiftRows.

The AddRoundKey is equivalent to itself.

The Number of Rounds

The number of rounds Nr depends on the key size

Key Size	Rounds
128	10
192	12
256	14

The Key Schedule

The key schedule takes an Nk-word key ($32 \cdot Nk$ bits) and computes the $Nr+1$ subkeys of a total size of $4(Nr+1)$ words.
Let `W` be a word array of size `W[4*(Nr+1)]`. The first 4 words of `W` become the subkey K_0, the next 4 words become K_1, etc.
The key schedule is as follows

```
W[0..Nk-1] = key[*];
for i := Nk to 4*(Nr+1)-1 do {
  temp = W[i-1]
  if((i%Nk)==0)
    temp = bytesubstitution(temp<<<8) ^ RCON[i/Nk];
  if((Nk==8) & ((i%Nk)==4))
    temp = bytesubstitution(temp);
  W[i] = W[i-Nk] ^ temp;
}
```

The Key Schedule (cont.)

where

- `bytesubstitution()` apply the S-box (from SubBytes) of Rijndael on each byte of `temp<<<8`, and

- `RCON` is fixed by

  `word32 RCON[] = {`
  $01_x,\ 02_x,\ 04_x,\ 08_x,\ 10_x,\ 20_x,\ 40_x,\ 80_x,$
  $1b_x,\ 36_x,\ 6c_x,\ d8_x,\ ab_x,\ 4d_x,\ 9a_x,\ 2f_x$
  `};`

  (these are powers of 2 in $GF(2^8)$).

Efficient Implementations

Efficient implementations of Rijndael combine the SubBytes, ShiftRows and MixColumns operations together.
The MixColumns operation can be performed by four table lookups (8-bit indices, 32-bit output), and XORing the results.
The ShiftRows operation can be eliminated by carefully indexing the indices of the input bytes to the succeeding Mix Column operation.
As the SubBytes operation is also implemented as table lookups, the three operations together can be implemented by 16 table lookups in total (and 12 XORs), using 4 precomputed tables of

$$\text{MixColumns}(\text{ShiftRows}(\text{SubBytes}(\cdot)))$$

Security

There are no complementation properties, nor equivalent keys.

The security of the cipher highly depends on the choice of the S boxes. If the S boxes were affine, the whole cipher would be affine, thus easily breakable. (linear: $f(x) = ax$; affine: $f(x) = ax + b$).

The best theoretic attack breaks up to 7 rounds with over 2^{110} complexity for 128-bit keys, and 2^{204} for 8-round version of the 256-bit keys.

No shortcut attacks on the full Rijndael are known.*

The Square Attack

Observe the following property:
If one byte is modified in the plaintext, then exactly 4 are modified after one round, and all the 16 are modified after two rounds.
The same property holds in the decryption direction.
Thus, a one-byte difference cannot lead to a one-byte difference after three rounds.
This observation can be used for an attack on 5 rounds of Rijndael. Using a few more tricks, up to 7 rounds can be attacked.