1.

Block Cipher and Stream Cipher are the methods used for converting the plain text into cipher text directly and belong to the family of symmetric key ciphers.
A block cipher is a deterministic and computable function of k-bit keys and n-bit (plaintext) blocks to n-bit (ciphertext) blocks. (More generally, the blocks don't have to be bit-sized, n-character-blocks would fit here, too). This means, when you encrypt the same plaintext block with the same key, you'll get the same result. (We normally also want that the function is invertible, i.e. that given the key and the ciphertext block we can compute the plaintext.)

To actually encrypt or decrypt a message (of any size), you don't use the block cipher directly, but put it into a mode of operation. The simplest such mode would be electronic code book mode (ECB), which simply cuts the message in blocks, applies the cipher to each block and outputs the resulting blocks. (This is generally not a secure mode, though.)

Some early encryption schemes like the one used by Caesar could be categorized as a "block cipher with 1-character blocks in ECB-mode". Or generally, everything that has a code book.

We usually use other modes of operation, which include an initialization vector and some kind of feedback, so that every block of every message is encrypted a different way.

A stream cipher is a function which directly maps k-bit keys and arbitrary length plaintexts to (same arbitrary length) ciphertext, in such a way that prefixes of the plaintext map to prefixes of the ciphertext, i.e. we can compute the starting part of the ciphertext before the trailing part of the plaintext is known. (Often the message sizes might be limited to multiples of some "block size", too, but usually with smaller blocks like whole bytes or such.)

If a part of the plaintext repeats, the corresponding ciphertext usually is not the same – different parts of the message will be encrypted in different ways.

Often such stream ciphers work by producing a keystream from the actual key (and maybe an initialization vector) and then simply XOR-ing it with the message – these are called synchronous stream ciphers. Other stream ciphers might vary the encryption of future parts of the message depending on previous parts.

Some block cipher modes of operation actually create a synchronous stream cipher, like CTR and OFB mode.

The major difference between a block cipher and a stream cipher is that the block cipher encrypts and decrypts a block of the text at a time. On the other hand, stream cipher encrypts and decrypts the text by taking the one byte of the text at a time.

Stream Cipher typically encrypts one byte of the message at that moment instead of using blocks.

Stream ciphers belong to the family of symmetric key ciphers.

Typically, single bits/bites are used as single digits. To avoid security concerns, it should be made sure that the same starting state is not used more than once.
Most widely used stream cipher is RC4.

a)

فایل‌های نظیر هر کدام از روش‌های RC2 و RC5 به همراه این فایل ارسال شد. پس از رمزنگاری توسط RC2 نرم‌افزار Cryptool موفق به شکستن رمز شد ولی برای RC4 موفق نشد

b)

برای روش‌های CBC DES و CBC 3DES موفق به شکستن رمز نشد (زمان تخمینی برای شکستن رمز بیش از 1000 سال تخمین زده شده است)

2.

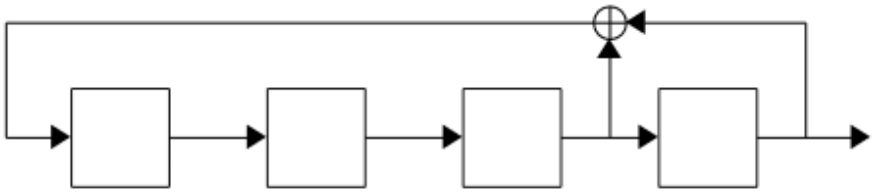
LFSRs come in three flavors:

LFSRs which generate a maximum-length sequence. These LFSRs are based on primitive polynomials.

LFSRs which do not generate a maximum-length sequence but whose sequence length is independent of the initial value of the register. These LFSRs are based on irreducible polynomials that are not primitive. Note that all primitive polynomials are also irreducible.

LFSRs which do not generate a maximum-length sequence and whose sequence length depends on the initial values of the register. These LFSRs are based on reducible polynomials.

x^4+x+1 ::

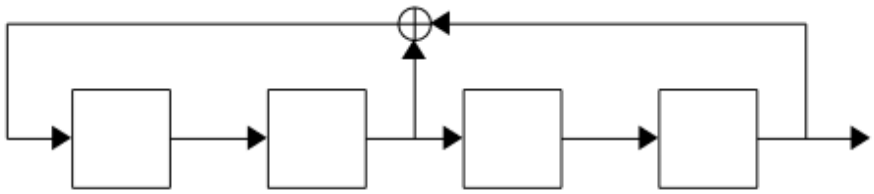


Duo to the table of states ::

	s_3	s_2	s_1	s_0	Output
1	0	0	0	1	1
2	1	0	0	0	0
3	0	1	0	0	0
4	0	0	1	0	0
5	1	0	0	1	1
6	1	1	0	0	0
7	0	1	1	0	0
8	1	0	1	1	1
9	0	1	0	1	1
10	1	0	1	0	0
11	1	1	0	1	1
12	1	1	1	0	0
13	1	1	1	1	1
14	0	1	1	1	1
15	0	0	1	1	1
	0	0	0	1	1

this is a primitive.

x^4+x^2+1



Duo to the initial state of registers we got 3 table ::

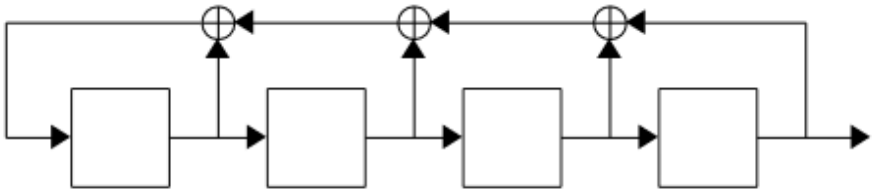
	s_3	s_2	s_1	s_0	Output
1	0	0	0	1	1
2	1	0	0	0	0
3	0	1	0	0	0
4	1	0	1	0	0
5	0	1	0	1	1
6	0	0	1	0	0
	0	0	0	1	1

	s_3	s_2	s_1	s_0	Output
1	1	0	0	1	1
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	1
5	0	1	1	1	1
6	0	0	1	1	1
	1	0	0	1	1

	s_3	s_2	s_1	s_0	Output
1	0	1	1	0	0
2	1	0	1	1	1
3	1	1	0	1	1
	0	1	1	0	0

therefor this is a reducible.

$x^4+x^3+x^2+x+1$



we got the table ::

	s_3	s_2	s_1	s_0	Output
1	0	0	0	1	1
2	1	0	0	0	0
3	1	1	0	0	0
4	0	1	1	0	0
5	0	0	1	1	1
	0	0	0	1	1

	s_3	s_2	s_1	s_0	Output
1	1	1	1	1	1
2	0	1	1	1	1
3	1	0	1	1	1
4	1	1	0	1	1
5	1	1	1	0	0
	1	1	1	1	1

	s_3	s_2	s_1	s_0	Output
1	1	0	0	1	1
2	0	1	0	0	0
3	1	0	1	0	0
4	0	1	0	1	1
5	0	0	1	0	0
	1	0	0	1	1

so this is a ireducible.

3.

a) 2 * 256

با داشتن این تعداد زوج متوالی میتوان 2 * 256 کلید متوالی را به دست آورد که در نتیجه میتوان با استفاده از آن یک معادله ی 256*2 مجهوله را حل کرد و با استفاده از مقادیر pin های مربوط به LSFR را به دست آورد

b)

رابطه ی بین LFSR و مقادیر S و مقادیر P را میتوان به شکل زیر نوشت

$$
\begin{array}{lll}
i=0, & s_m & \equiv p_{m-1}s_{m-1}+\ldots+p_1s_1+p_0s_0 \mod 2 \\
i=1, & s_{m+1} & \equiv p_{m-1}s_m+\ldots+p_1s_2+p_0s_1 \mod 2 \\
\vdots & \vdots & \vdots \\
i=m-1, & s_{2m-1} & \equiv p_{m-1}s_{2m-2}+\ldots+p_1s_m+p_0s_{m-1} \mod 2
\end{array}
$$

بنابراین نهایتاً یک معادله داریم که با داشتن 2m از جواب ها(مقادیر S0 تا S2m-1) دارای m مجهول(p0 تا pm-1) می‌شویم و میتوان این m معادله و m مجهول را حل کرد.

c)

در این سیستم در هر کلاک یک بیت به عنوان خروجی داده می‌شود که یک بیت از کلید است. چون مقدار اولیه ی LFSR ممکن است خود دارای خواص آماری مناسب نباشد و حالت psudo random بودن را ندارد بنابراین نمیتوان از آن به عنوان قسمتی از کلید استفاده کرد. همچنین چون ممکن است طول آن کمتر از طول کلید مورد نیاز ما باشد نمیتوان از آن استفاده کرد.

4.

یکی از مهمترین خواص S-Box ها که باعث می‌شود که DES ایمن باشد تحلیل خطی این S-BOX هاست به صورتی که رابطه ی ریاضی نظیر هر کدام یک رابطه ی غیرخطی است که باعث می‌شود حل آن و به دست آوردن وارون آن یک مسأله ی NP باشد.

s1(x1) = 13	s1(x2) = 04	s1 xor s2 = 09	s1(x1 xor x2) = 08
s2(x1) = 09	s2(x2) = 00	s1 xor s2 = 09	s1(x1 xor x2) = 05
s3(x1) = 12	s3(x2) = 13	s1 xor s2 = 01	s1(x1 xor x2) = 01
s4(x1) = 14	s4(x2) = 10	s1 xor s2 = 04	s1(x1 xor x2) = 09
s5(x1) = 03	s5(x2) = 04	s1 xor s2 = 07	s1(x1 xor x2) = 06
s6(x1) = 13	s6(x2) = 09	s1 xor s2 = 04	s1(x1 xor x2) = 08
s7(x1) = 12	s7(x2) = 01	s1 xor s2 = 13	s1(x1 xor x2) = 06
s8(x1) = 11	s8(x2) = 07	s1 xor s2 = 12	s1(x1 xor x2) = 02

5.

a)

با توجه به ساختاری که در کتاب معرفی شده است در مرحله ی IP بیت شماره 57 به بیت 33 مپ میشود. یعنی در مرحله ی اول Feistal Network خود داریم L0 = 0 و $R0 = 2^{31}$

بنابراین در f(R0) در E(R0) که عملیات expand انجام می‌شود بیت 1 را به بیت 2 و 48 مپ میکند و بنابراین ورودی S1 برابر 010000 و ورودی S8 برابر 000001 می‌شود و بقیه ی S-box ها ورودی صفر دارند.

b)

باتوجه به ساختارطراحی Des با تغییر این یک بیت پس از یک راند حداقل 2 بیت تغییر میکند.

c)

با توجه به ورودی S-box هه که در قسمت a به دست آمد خروجی S-box ها به شکل زیر است

S1 = 0011
S2 = 1111
S3 = 1010
S4 = 0111
S5 = 0010
S6 = 1100
S7 = 0100
S8 = 0001

بنابراین پس از مرحله ی Mapping که پس از s-box ها انجام می‌شود خروجی به شکل 11010000010110000101101110011110 خواهد بود. نهایتاً این خروجی با L0 باید xor شود که قبلاً گفتیم L0 برابر 0 است. پس نهایتاً R1 برابر رشته ی بالا خواهد شد و L1 = R0 خواهد بود

d)

با توجه به قسمت c و قسمت a نهایتاً پس از یک راند 6 بیت متفاوت خواهد بود با حالتی که تمام ورودی صفر باشد

6.

a)

با توجه به اینکه ما 16 راند داریم و نیاز به 16 زیر کلید داریم شرط زیر باید برای زیر کلید ها برقرار باشد

$ki+1=k16-i$ for $i=0,1,...,7$

که با توجه به ساختار ایجاد زیر کلید ها میتوان این رابطه را باز تر نیز نوشت

b)

با توجه به ساختار rotate مانندی که در هر مرحله روی کلید انجام می‌شود تنها کلید هایی که باعث می‌شود شرایط بالا رعایت شود کلید هایی است که در هر نصف ی خود تنها دارای صفر یا یک باشند که با rotate شدن در هر مرحله تغییر نکنند. بنابراین چهار کاندید زیر را (در هکس) داریم

00000000000000
0000000FFFFFFF
FFFFFFF0000000
FFFFFFFFFFFFFF

c)

با توجه به اینکه 4 کلید ضعیف تنها وجود دارد، احتمال انتخاب رندوم یکی از این چهار کلید برابر $4/2^{56}$ است که برابر $1/2^{54}$ است