Chapter 2
Symmetric Encryption Algorithms

February 15, 2010

2

The term "symmetric" means that the same key used to encrypt is used decrypt.

In the widest sense all pre-PKC encryption algorithms are symmetric, although their keys may be very different in form. For example:

Caesar's algorithm; a one character key:

Vigenère and Beaufort algorithms; a string of characters sequentially repeated form the key.

Wheatstone, Jefferson and even the Enigma machines;the key is a selection of discs (labelled or with electrical connections) which may be periodically changed.

But with the advent of the computer as a normal component of storage and transmission systems, the key is now a string of bits. Originally this might be 64 bits. Nowadays 128, 192 or 256 bits are considered safer. The string should be pseudo-random, and its length m such as to make "try all the 2^m keys" or "brute force" attacks infeasible. (Such an attack tries to encrypt known or guessed plain-text and see if some key produces observed cypher-text; or to decrypt known cypher-text and see if plausible plain-text results.)

The algorithms to which such keys apply are typically block cypher: or algorithms which, under key control, map n bits of input into n bits of output. They are, of course, reversible - for decryption. Typically n is 64, 128, 192 or 256 bits. A good algorithm provides confusion and diffusion.

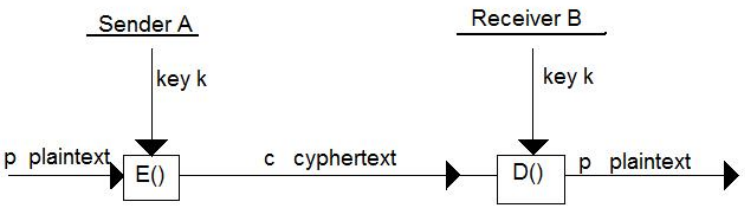
Confusion messes up the input bit pattern, typically by substituting one bit pattern by another, or permuting bits - always in a reversible manner.

Diffusion causes a change in one part of the input plain-text or key to spread out and make changes throughout the output cypher-text. Changing

a single key bit or plain-text bit should change 50% of all cypher-text bits in a "random" fashion.

Knowledge of c or p, without knowing k, should reveal nothing about p or c respectively. Knowledge of p and c should reveal nothing about k.

The basic symmetric scheme is:



We write this $C = E(k;\ p)$ $p = D(k;\ c)$ with E=Encryption, D=Decryption (E^{-1} sometimes is used).

2.1 DES The Data Encryption Standard (Algorithm)

DES, introduced in the 1970's, is the most widely used encryption algorithm, although it is no longer safe. (Theoretically the key, if unchanged, can be found if one has enough - a very large number - of plain-text cypher-text pairs. In practice this is seldom a practical method of attack.)

The full DES specification is an American National Standard and can be found at: http://www.ii.uib.no/ osvik/des/fips46-3.pdf

A summary only is presented here:

DES has a 64-bit key, of which only 56 bits are used. DES has 64 input bits (plain-text) and 64 output bits (cypher-text). It has sixteen stages or "rounds" and uses a Feistel network, in which the 64 bits af data are split into two 32-bit blocks, left (L) and right (R). In round n we have, with $n = 1$ *to* 16. See figure 2.1.

$$L_n = R_{n-1} \quad R_n = L_{n-1} \oplus f(k_n, R_{n-1})$$

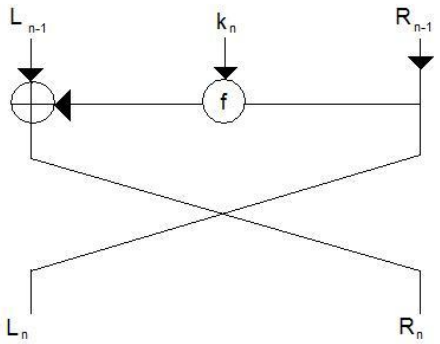
This process is reversible: Knowing $L_n,\ R_n$ and k_n we can find L_{n-1} and R_{n-1}. The round sub-key k_n is derived from the basic key k by a simple key-generation procedure. The function $f()$ is the core of DES. It is as follows, see figure 2.2:
The 32-bit input (R_{n-1}) is expanded to 48 bits. This is XORed with the round key k_n. The result is split into eight 6-bit blocks presented to eight S-boxes (substitution boxes).

Each S-box produces a 4-bit output so that we are back to 32-bits, which are then permuted by a fixed function P. The S-boxes are non-linear $S(x \oplus y) \neq S(x) \oplus S(y)$ where $x,\ y$ are possible 6-bit inputs; but are uniform in the sense that each 4-bit output pattern appears four times to account for the 64 6-bit input patterns. (But the S-boxes are differentially non-uniform, see module 1).

There is a 64-bit input permutation, IP, before the data enter the Feistel network, and IP^{-1} is applied to the final network output - to ensure correct decryption.

The following is one of 16 stages. (Figure 2.1):

The Feistel Network



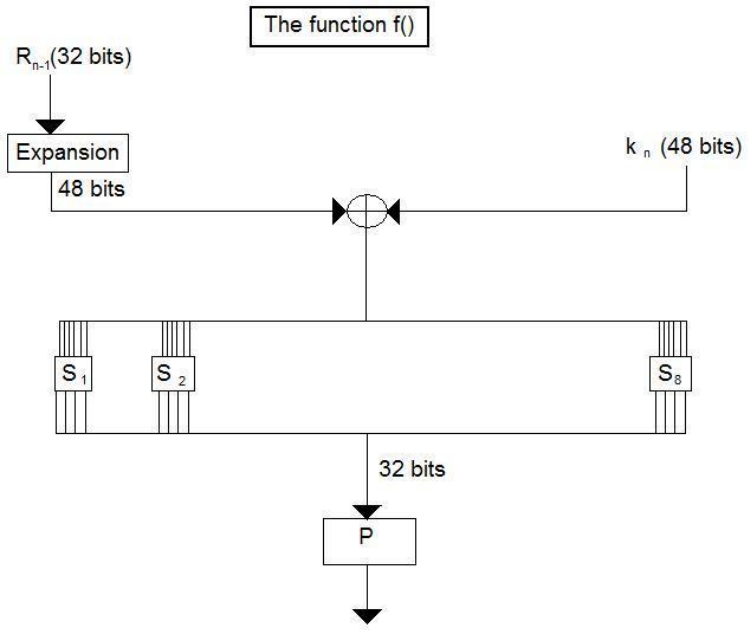


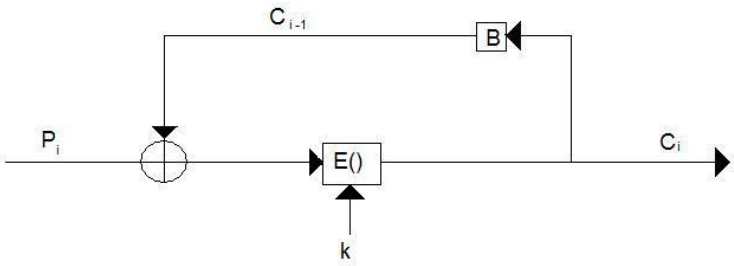
Figure 2.F2 The Function f()

2.2 Modes for Block Cyphers

Algorithms such as DES handle fixed length blocks (example: 64 bits). To encrypt longer streams of data various "modes" may be used. We call the ith input plain-text block p_i and the corresponding cypher-text block c_i. Some recognised modes are:

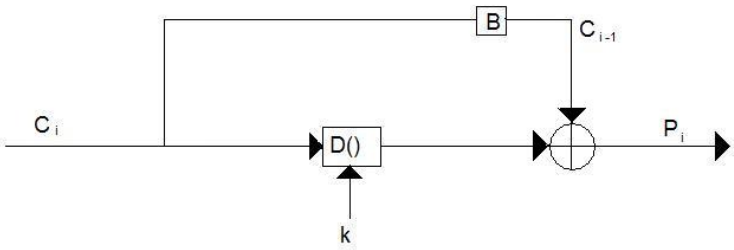
Electronic Codebook Lookup (ECB) $c_i = E(k; p_i)$. Weakness: If $p_i = p_j$ then $c_i = c_j$ showing up the identical plain-text.

Cypher Block Chaining (CBC)
$c_i = E(k; (p_i \oplus c_{i-1}))$



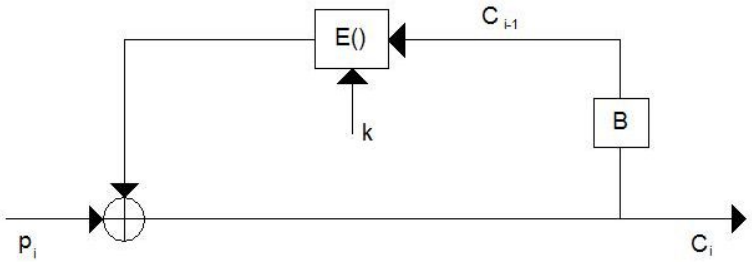
B is a buffer store. The addition is bit-by-bit XOR.
Decryption is

$p_i = D(c_i) \oplus c_{i-1}$



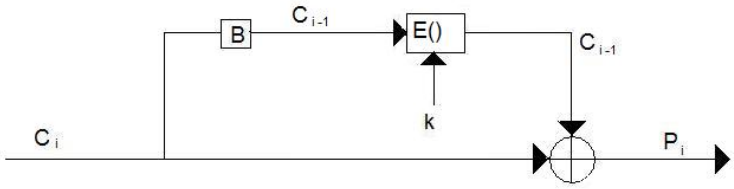
Note that the buffer must be initialised with an IV or Initial Value when $i = 1$.

Cypher Feedback (CFB)



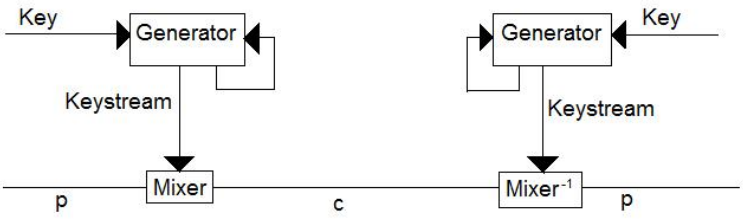
$c_i = p_i + E(k; c_{i-1})$

Decryption is



$p_i = c_i \oplus E(k, c_{i-1})$
Note here that there is no reason why $E()$ should be reversible. No decryption applies, and $E()$ could be any key controlled scrambler.

Other Methods There are several other methods of using a block algorithm to encrypt a long stream of data. The feedback-on-encryption approach (feed-forward-on-decryption) can be replaced by feed-forward-on-encryption (feedback-on-decryption) but this is not advisable; because one corrupted cypher-text block will perpetuate the corruption of decrypted plain-text indefinitely. Another approach is to have a key controlled "white-noise" key-stream generator at each end "mixing" with the plain-text to produce cypher-text. If the mixer is simple XOR than the decryption is identical; otherwise the mixer needs to be reversible - for example an encryptor/decryptor pair with the generator a changing key source.



Clearly if the mixer is too simple the system is susceptible to an attack to find the key-stream from known plain-text and cypher-text. An example of such an encryption system is A5/1, see appendix 2.X.

2.3 The AES Project

64-bit DES (56-bit key) was perceived to be weak. One method of strengthening its use is Triple-DES.

$$c = E(k_1;\ D(k_2,\ E(k_1;\ p)))$$
$$p = D(k_1;\ E(k_2;\ D(k_1;\ (c))))$$

With two 56-bit keys k_1 and k_2 the security is greatly increased, although the data per block is still only 64-bits, so the computational speed is divided by three.

A move was launched in the USA to develop a DES-replacement, faster and more secure; the increased security being essentially supplied by longer keys (128,192 or 256 bits) and increased resistance to Differential and Linear Crypt-Analysis, through Pre- and Post- whitening, and more uniform differential and linear characteristics. Pre and post whitening, achieved by XORing the plain-text and cypher-text with the first and last pseudo-random sub-keys stops the attacker seeing the absolute values of these. The new standard was to be designated AES, the Advanced Encryption Standard.

An AES call for proposals was issued and from the 20-odd submissions a shortlist of five was selected. The short list was:

1. SERPENT (UK, Israel, Norway) 32-round using 32 4 by 4 S-boxes
2. RC6 (USA) 20-round, 2-modified, 2-unmodified Feistel network with complicated functions
3. Rijndael (Belgium) The winner, see below
4. MARS (IBM) 32-round Feistel network, with eight initial and eight final rounds of mixing and 16 central rounds using sub-keys
5. Twofish (USA) 16-round 2-modified, 2-unmodified Feistel network with complicated functions.

All contenders could handle the 128, 192 or 256 bit key requirement, and had pre- and post-whitening.

The winner Rijndael works with a $4*4$ (16-byte) array of data (in the 128-bit version) into which plain-text is placed, modified on its own and with sub-keys, and from which the cypher-text is read. The original specification is to be found at http://www.techheap.com/cryptography/encryption/spec.v36.pdf. The following description is somewhat abbreviated.

2.4 Rijndael

Authors: Joan Daemen (Banksys, Brussels, Belgium), Vincent Rijmen (KU Leuven, Belgium)
Technique: 'Squares', with non-linear S-boxes, row rotations, and column mixing - all reversible.
Number of rounds:10 (128-bit user key), 12(192-bit user-key), 14(256-bit user-key).
Sub-Key: One more than the number of rounds (i.e. 11, 13 or 15) 128-bit sub-keys.

General Comment: The general definition of Rijndael caters for data block lengths of 128,192 or 256 bits, but we only consider 128 bits here.

In Rijndael, bytes may be considered as elements in the field $GF(2^8$, defined by the irreducible polynomial over $GF(2)$ $z^8 + z^4 + z^3 + z$. Their values are represented in hex, example: 'C5'. A 32-bit word is sometimes considered to be a polynomial of degree 3 in x, with four byte coefficients in this $GF(2^8)$.

2.4.1 Encryption

The basic square matrices of data and sub-keys are $4 * 4$, the entries being bytes $a(i,\ j)$, $k(i,\ j)$. The plain-text is written in initially as $a00$, $a10$, $a20$, $a30$, $a11\ldots$ etc. and after the final round the cypher-text is read out similarly. The current content of the data matrix is referred to as the 'state'.

There is a pre-whitening step, $XK1$, with the matrix of the first sub-key XORed into the plain-text state, byte by byte.

Each round then consists of four steps:

1. BS: Byte substitution. Each 8-bit byte in the state is reversibly mapped into another byte (S-box)
2. SL: Shift-row-left. Row 0 is unmoved, row 1, row 2 and row 3 are rotated left 1, 2 and 3 bytes respectively.
3. MC: Mix-column. Each column is regarded as a degree-3 polynomial over $GF(2^8)$, and multiplied by the polynomial $p(x) =' 03' x^3 +' 01' x^2 +' 01' x +' 02'$, modulo $(x^4 + 1)$.
4. XKi: Sub-key addition. The sub-key for the current round is XORed into the state.

The last round is an exception, since the mix-column operation is omitted. Its last sub-key addition, XK11, is a post-whitener.

The purpose of the pre- and-post whiteners is to prevent an attacker from knowing the input to the first or output from the last round, thus making differential crypt-analysis in the reverse of forward direction more difficult.

The BS S-box can be a look-up table with 256 8-bit entries. Alternatively it can be implemented by: replacing each byte by its multiplicative inverse in $GF(2^8)$. The 'inverse' of '00' is itself. Applying a defined reversible affine transformation to the result (over $GF(2)$).

The S-box provides non-linear transformations at the byte level. The shift-row and the mix-column ensure diffusion. For efficient implementation

on 32-bit processors, the four steps of a round can be combined into a single set of table look-up, byte by byte. The combined BS, SL, MC operations on a single byte in the ith row produce a new 32-bit value for the column i places to the left. When all 16 bytes have been so processed the new contents of a column is the XOR (linear superposition) of four such 32-bit values arising from bytes in row i, i places to the right. This suggests the possibility of a differential attack, exploiting any non-uniform differential characteristic in these alternative four 8-by-32-bit S-boxes, since the superposition can be neutralised. A linear crypt-analysis attack would appear to be harder. However, both attacks assume the existence of non-uniform characteristics, and presumably the authors of Rijndael have ensured that the BS and MC (and their inverses) do not have such characteristics - or if they do, then with extremely low probability.

(Yet another way of looking at Rijndeal is to view each round as the application of a single 32-by-32-bit S-box to $a(0,\ i),\ a(1,\ i+1),\ a(2,\ i+2),\ a(3,\ i+3)$) (with subscripts modulo 4) for $i = 0\ to\ 3$; followed by the XOR of the round key.)

2.4.2 Decryption

This is the reverse of encryption; i.e. with rounds and sub-keys in reverse order, and the operations within each round inverted and in inverse order. The inverse of the S-box can be in table form; alternatively the affine transformation and multiplicative inverses can be inverted. The inverse of the mix-column is got by multiplying by the inverse of $p(x) : (p(x))^{-1}$.

$$(p(x))^{-1} = `OB'x^3 + `OD'x^2 + `09'x + `OE'$$

Moreover a little thought shows that the structure (the sequence of operations) of decryption can be made the same as that of encryption (using the inverse functions rather than the original ones), provided that the sub-keys have themselves been subjected to the mix-column operation as well as being used in reverse order. This is because shift-row and byte-substitution commute, and so do mix-column and sub-key addition if the sub-keys are thus transformed. Using this same structure for decryption as for encryption permits fast and compact implementation.

To see this we write encryption as:

XK1, for i=2 to 10 (BS, SL, MC, XKi), BS, SL, XK11 (0.1)

With $XK^{-1} = XKi,\ SL^{-1} = SR,\ MC^*,\ BS^{-1} = BS^*$ for short; the formal inverse of equation **??** for decryption is:

$XK11,\ SR,\ BS^*$ for $i = 10\ to\ 2\ (XKi,\ MC^*,\ SR,\ BS^*),\ XK1$, which using the commutition becomes

$XK11,\ BS^*,\ SR,\ MC^*,\ MK^*10$ for $i = 9\ to\ 2\ (BS^*,\ SR,\ MC^*,\ XKi),\ BS^*,\ SR,\ XK1$ $= XK11,\ for\ i = 10\ to\ 2\ (BS^*,\ SR,\ MC^*,\ XKi^*),\ BS^*,\ SR,\ XK1$

This is the same structure as 1.1, but with the key schedule reversed; with the inverse of BS, SL and MC; and using XKi*, which is XKi subjected to multiplication by the inverse polynomial $(p(x))^{-1}$.

2.4.3 The Key Schedule

The number of 32-bit words in the user-supplied key is denoted by Nk, so $Nk = 4,\ 6,\ or\ 8$ for 128, 192 or 256-bit user keys. The user key is expanded to 11, 13 or 15 128-bit (4 word) sub-keys, respectively. There is one expansion rule if $Nk = 4\ or\ 6$, another if $Nk = 8$. In all cases the expanded key schedule is a linear list of 4-byte words $w(j)$ with $j = 0\ to\ 4$. Nr (Nr=Number of rounds=10, 12 or 14); and the first Nk words are just the user key. Thus for $Nk = 4$, the first sub-key is the user key; for $Nk = 6$ the first sub-key and half the second-sub-key are the user key; for $Nk = 8$ the first two sub-keys are the user keys.

For $Nk = 4\ or\ 6$ the expansion rule is:

for $j = Nk,\ 2Nk, 3Nk. \ . \ .4Nr$
$w(j) = w(j - Nk) \oplus SB(Rotl(w(j - 1))) \oplus Rcon(j/Nk)$
for $i = 1$ to $Nk - 1$ $w(j + 1) = w(j + i - Nk) \oplus w(j + i - 1)$

Where $Rotl()$ is a left-byte-rotate of a word, $SB()$ is a four-byte substitution look-up, and $Rcon()$ is a round constant.

For $Nk = 8$ the expansion rule is:

for $j = Nk,\ 2Nk, 3Nk. \ . \ .4Nr$

$w(j) = w(j - Nk) \oplus SB(Rotl(w()j - 1)) \oplus Rcon(j/Nk)$
for $i = 1,\ 2,\ 3 \quad w(j + i) = w(j + i - Nk) \oplus w(j + i - 1)$
$w(j + 4) = w(j + 4 - Nk) \oplus SB(w(j + 3))$
for $i = 5,\ 6,\ 7 \quad w(j + i) = w(j + i - Nk) \oplus w(j + i - 1)$

Note that in the about routines $(i + j)$ must not exceed $4(Nr + 1)$. Note also that not only does the proposed key schedule use the user-supplied key 'in clear', but that the whole schedule is 'one pass', allowing the sub-keys to be calculated on the fly with minimal storage requirements.

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

x^0	a_{00}	a_{01}	a_{02}	a_{03}
x^1	a_{10}	a_{11}	a_{12}	a_{13}
x^2	a_{20}	a_{21}	a_{22}	a_{23}
x^3	a_{30}	a_{31}	a_{32}	a_{33}

Appendix X

The A5/1 encryption algorithm is used in GSM digital mobile telephony. GSM telephony contains various security functions:

1. User Authentication
2. User Confidentiality
3. Signalling Confidentiality
4. Data Confidentiality

Three algorithms A3, A5 and A8 are used. The data (digital voice in the form of 114-bit frames transmitted at 115 frames/second or 13000 bit/second) are encrypted using A5. A5 is a pseudo random number generator and an XOR mixer with the data stream.

A5 uses three linear feedback shift registers (see Appendix M10.Y) of lengths 19, 22, 23 based on three primitive polynomials $(x^{19} + x^5 + x^2 + x + 1)$, $(x^{22} + x + 1)$, $(x^{23} + x^{16} + x^2 + x + 1)$ respectively. There are 2^{64} initial states possible, and since the registers generate maximum length sequences a very large number of shifts is required before they jointly return to their initial states.

Each register has a clock bit (corresponding to x^{11}, $x^{12} and x^{13}$ respectively). As each data bit is processed it is XORed with the 3-bit corresponding to x^0 in the registers, and then each register is rotated one bit provided that its clock bit value agrees with the majority value over all three clock bits. As a result the probability of a register being shifted is $\frac{3}{4}$.

When a frame of 114 bits is to be encrypted each register is initialised as follows:

1. Clock in the session key in parallel 64 times. (The session key is established when the call in question is set up). The stuttering stop/go shifting does not apply.
2. Clock in the 22-bit Frame Number (within the call) in parallel. The stop/go does not apply. We have now reached the initial state.
3. Then clock the register 100 times with the stop/go mechanism activated
4. Then clock 228 bits further (one frame in each direction) and XOR the three outputs with the data.

The security of A5 has been shown to be not very good - always assuming an attacker can observe the key stream (the XOR of the three register outputs), which is not very plausible. Nevertheless it has been suggested that a change in the choice of clock bits would be better.

Appendix Y
Binary Linear Feedback Shift Registers for Pseudo-Random Bit Streams

1. Polynomials over $GF(2)$ *modulo* $p(x)$, where $p(x)$ is irreducible form a field. Obviously multiplication, addition and subtraction hold. To see that elements have an inverse we use Euclid. Let $p(x)$ have degree of m, $r(x)$ have degree less than m, then we can find:

$$a(x)p(x) + b(x)r(x) = h(x) = HCF(p(x),\ r(x)) = 1$$

$$so\ r(x)^{-1}\ mod\ p(x) = b(x)$$

Example $p(x) = x^5 + x^2 + 1 \quad r(x) = x^3 + x^2 + 1$
Divide: $(x^5+x^2+1) \div (x^3+x^2+1) = (x^2+x+1)$ with remainder (x^2+x)

$$\begin{aligned} So\ (\mathrm{x}^5 + x^2 + 1) &= (\mathrm{x}^3 + x^2 + 1)(x^2 + x + 1) + (x^2 + x) \\ (\mathrm{x}^3 + x^2 + 1) &= \mathrm{x}(\mathrm{x}^2 + x) + 1 (= HCF) \\ Therefore\ 1 &= (\mathrm{x}^3 + x^2 + 1) + x((x^5 + x^2 + 1) + (x^3 + x^2 + 1)(x^2 + x + 1)) \\ &= (\mathrm{x}^3 + x^2 + 1)(1 + x^3 + x^2 + x) + x(x^5 + x^2 + 1) \\ So \qquad & (\mathrm{x}^3 + x^2 + 1)^{-1}\ mod\ (x^5 + x^2 + 1) = (x^3 + x^2 + x + 1) \end{aligned}$$

2. There are 2^m polynomials in this field, and the $2^m - 1$ non-zero polynomials form a multiplicative group of order $(2^m - 1)$. Therefore the order of any element divides $(2^m - 1)$.

   In our example $m = 5,\ 2^m - 1 = 31 = a\ prime$, therefore all elements have order=31.

3. In general we may consider the field $GF(2^m)$ composed either of polynomials in x or as polynomials in α where α is a roof of $p(x)$ (so $p(\alpha) = 0$ equivalent to the $mod\ (p(x))$ operation). If α is a root so are $\alpha^2,\ \alpha^4,\ \alpha^8 \ldots \alpha^{2^{m-1}}$ because

   $$p(\alpha) = 0\ \rightarrow\ (p(\alpha))^2 = 0\ \rightarrow\ p(\alpha^2) = 0$$

   over $GF(2)$. Thus $p(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^4) \ldots (x + \alpha^{2^{m-1}})$.

4. If α has $\Theta(\alpha) = s$ then $\alpha^s = 1$ and $s|(2^m - 1)$. So $x^{2^m-1} = 1$ and all the roots of $p(x)$ have orders which are the same and divide $2^m - 1$. Therefore $p(x)|(x^{2^m-1} + 1)$. If $s = 2^m - 1$ then α is called primitive and $p(x)$ is called a primitive polynomial. If $s \neq 2^m - 1$ then α is non-primitive and $p(x)$ is a non-primitive polynomial.

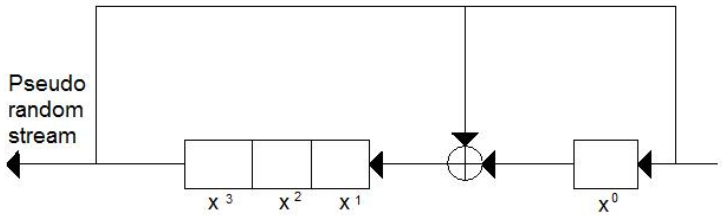
5. It can be proved that there always exists a primitive element in the field and therefore a primitive polynomial.

   Example 1. $x^4 + x + 1$ is primitive and defines a representation of $GF(2^4)$ with elements

$$1,\ \alpha,\ \alpha^2,\ \alpha^3,\ \alpha^4 = \alpha + 1,\ \alpha^5 = \alpha^2 + \alpha, \alpha^6 = \alpha^3 + \alpha^2,\ \alpha^7 = \alpha^4 + \alpha^3 = \alpha^3 + \alpha + 1, \alpha^8 = \alpha^4 + \alpha^2 + \alpha = \alpha^2 + 1,\ \alpha^9 = \alpha^3 + \alpha, \alpha^{10} = \alpha^4 + \alpha^2 = \alpha^2 + \alpha + 1,\ \alpha^{11} = \alpha^3 + \alpha^2 + \alpha, \alpha^{12} = \alpha^4 + \alpha^3 + \alpha^2 = \alpha^3 + \alpha^2 + \alpha + 1, \alpha^{13} = \alpha^4 + \alpha^3 + \alpha^2 + \alpha = \alpha^3 + \alpha^2 + 1, \alpha^{14} = \alpha^4 + \alpha^3 + \alpha = \alpha^3 + 1,\ (\alpha^{15} = \alpha^4 + \alpha = 1)$$

Example 2. $x^4+x^3+x^2+x+1$ is non primitive with the same elements but labelled differently. Let β be a root then we have $1,\ \beta,\ \beta^2,\ \beta^3,\ \beta^4 = \beta^3 + \beta^2 + \beta + 1,\ (\beta^5 = \beta^4 + \beta^3 + \beta^2 + \beta = 1)$. Therefore $\Theta(\beta) = 5/15$. β of this example corresponds to α^3 in the above example.

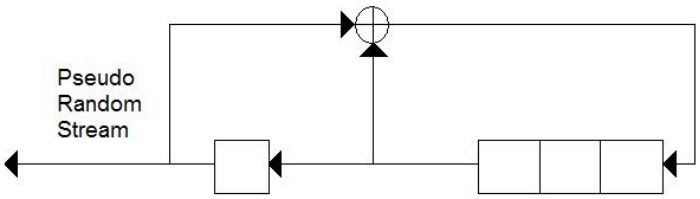
6. It is clear that successive powers of the roots of a primitive polynomial go through all the non-zero elements of the field. We may represent this process by a circuit.



Operation Xx (times x) is shift left. Modulo $p(x)$ is feed back, put 1 at the RH end and successive shifting gives 0001, 0010, 0100, 1000, 0011, 0110, 1100, 1011, 0101, 1010, 0111, 1110, 1111, 1101, 1001, (0001) (see example one above).

7. We can construct such feed back registers of length m for any primitive polynomial of degree m, and they will take $2^m - 1$ steps to repeat and reach their initial state again. Moreover since they go through all non-zero states, their output (LM end) is a pseudo-random stream of bits with 2^{m-1} 1's and $2^{m-1}-1$ 0's. It is called a maximum length sequence. Such circuits are used to produce "white noise".

8. Since $p(x)|(x^{2^m-1}+1)$ we have a quotient $q(x) = \frac{x^{2^m-1}+1}{p(x)}$. $q(x)$ has degree $(2^m - 1 - m)$. Expressed in terms of α (a root of $p(x)$) we have $p(\alpha)q(\alpha) = \alpha^{2^m-1}+1 = 0$! This gives an idea for another circuit multiplying $p(x)q(x)$ together to produce zero - or rather constructing $q(x)$ as we shift along in such a way to set the product to zero. We require the connections in the opposite direction to (6) above, example



we get 0001, 0010, 0100, 1001, 0011, 0110, 1101, 1010, 0101, 1011, 0111, 1111, 1110, 1100, 1000, (0001)

In this alternative representation it is easier to see the pseudo random bit stream emerging because it is simply the contents of the register, going through all possible non-zero values being shifted out.

9. GSM security algorithm A5 uses these ideas in its stream encryptor.