T-79.4501
Cryptography and Data Security

Lecture 4:
Stream ciphers (->Jan 31)
Block cipher confidentiality modes of operation
Stallings: Ch 6, Ch 3

1

Stream ciphers

- Stream ciphers are generally faster than block ciphers, especially when implemented in hardware.
- Stream ciphers have less hardware complexity.
- Stream ciphers can be adapted to process the plaintext bit by bit, or word by word, while block ciphers require buffering to accumulate the full plaintext block.
- Synchronous stream ciphers have no error propagation; encryption is done character by character with keys K_i that are independent of the data

  $C_i = E_{K_i}(P_i)$
- Function E is simple, the function which computes the key sequence is complex
- Example: Vigenère cipher, One Time Pad

  $C_i = (P_i + K_i) \bmod 26$

2

Stream cipher encryption

SENDER

(Secret key, Initial value) → Key stream

(Key stream , Message) → Ciphertext

RECEIVER

(Secret key, Initial value) → Key stream

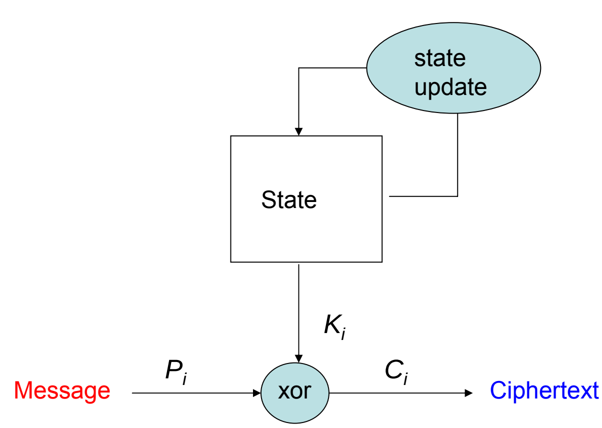
(Ciphertext, Key stream) → Message

The initial value can be public or secret, but it must not repeat during the lifetime of the secret key.

This is the operation of the basic, so called *synchronous stream cipher*

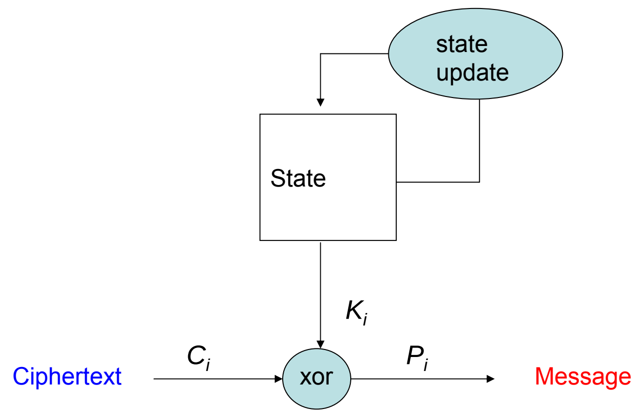
3

Synchronous stream cipher: encryption



4

Synchronous stream cipher: decryption



5

Stream ciphers: Security

- Known plaintext gives known key stream. Chosen plaintext gives the same but nothing more.
- Chosen ciphertext attack may be a useful method for analysing a self-synchronising stream cipher.
- The attacker of a stream cipher may try to find one internal state of the stream cipher to obtain a functionally equivalent algorithm without knowing the key.
- Distinguishing a key stream sequence from a truly random sequence allows also the keystream to be predicted with some accuracy. Such attack is also called prediction attack.

Requirements:

- Long period
- The initial state value can be public or secret, but it must not repeat during the lifetime of the secret key.
- Given a fixed initialisation value, the stream cipher generates a different keystream for each different key.

6

Stream ciphers: Designs

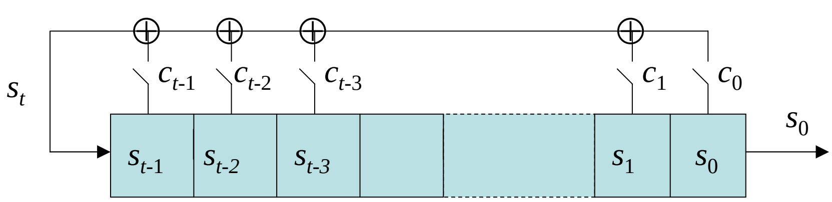
Linear feedback shift register (LFSR)

- LFSRs are often used as the running engine for a stream cipher.

Stream cipher design based on LFSRs uses a number of different LFSRs and nonlinear Boolean functions coupled in different ways. Three common LFSR-based types of stream cipher can be identified:

- *Nonlinear combination generators*: The keystream is generated as a nonlinear function of the outputs of multiple LFSRs
- *Nonlinear filter generators*: The keystream is generated as a nonlinear function of stages of a single LFSR.
- *Clock controlled generators*: In these constructions, the necessary nonlinearity is created by irregular clocking of the LFSRs. The GSM encryption algorithm A5/1 is an example of a stream cipher of this type.

Linear Feedback Shift Register (LFSR)



$$s_t = \sum_{i=0}^{t-1} c_i s_i = c_{t-1}s_{t-1} + c_{t-2}s_{t-2} + \ldots + c_0 s_0$$

The taps c_i are defined be giving the *feedback polynomial*

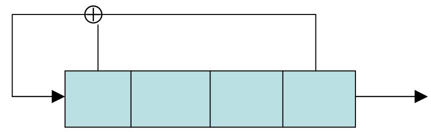
$$f(x) = x^t + c_{t-1}x^{t-1} + c_{t-2}x^{t-2} + \ldots + c_1 x + c_0$$

LFSR: Example

NOTE: Assume now that everything is binary, that is, in bits. Sums are taken mod 2. (Non-binary LFSRs exist.)

$f(x) = x^4 + x^3 + 1$

$\Rightarrow c_0 = c_3 = 1$ and $c_1 = c_2 = 0$



Let us take this as an initial state:	0	0	1	1
Then the next state is this:	1	0	0	1
And so on:	0	1	0	0
	0	0	1	0
	0	0	0	1
	1	0	0	0
For how long it goes?	...	...	...	...

LFSR statistical properties

A full cycle of 2^n -1 produces a sequence of length 2^n -1 (maximum length).

A maximum length sequence has ideal statistical properties:

– 2^{n-1} -1 zeroes and 2^{n-1} ones

– One string of ones of length *n;* one string of zeroes of length *n*-1

– Also ones and zeroes occur in about equally many pairs, triples ... , and so on.

A maximum length sequence (m-sequence) is achieved using a so-called primitive polynomials. For a source of primitive polynomials see: http://fchabaud.free.fr/English/default.php?COUNT=1&FILE0=Poly

Golomb's randomness postulates

R1: In the cycle of the sequence the number of 1-bits differs from the number of 0-bits by at most 1.

R2: In the cycle of the sequence, at least ½ of the runs have length 1, at least one ¼ have length 2, at least ⅛ have length 3, etc., as long as the number of runs so indicated exceeds 1. Moreover, for each of these lengths, there are (almost) equally many gaps and blocks.

R3: Let N be the length of the cycle (period) of the sequence (s_i).The *autocorrelation function* is two-valued. For some integer K:

$$C(k) = \frac{1}{N}\sum_{i=0}^{N-1}(2s_i - 1)(2s_{i+k} - 1) = \begin{cases} 1, \text{if } k = 0, \\ \frac{K}{N}, \text{if } 1 \le k \le N-1 \end{cases}$$

Definition: A binary sequence which satisfies Golomb's randomness postulates is called a *pseudo-noise* or a *pn-sequence.*

Example

Consider the sequence with cycle length 15:

`0 1 1 0 0 1 0 0 0 1 1 1 1 0 1`

R1: The number of 0-bits is 7, the number of 1-bits is 8

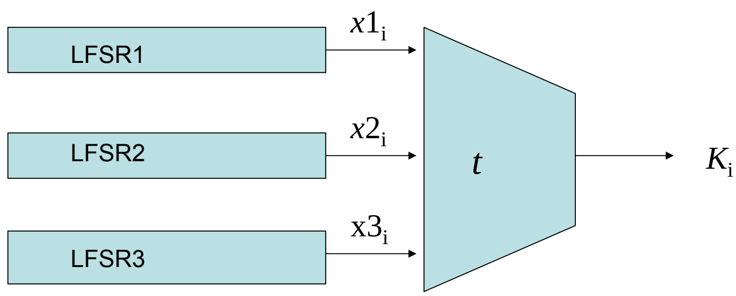
R2: the sequence has eight runs:

- 4 runs of length 1(2 gaps and 2 blocks)
- 2 runs of length 2 (1 gap and 1 block)
- 1 run of length 3 (1 gap)
- 1 run of lenght 4 (1 block)

R3: The autocorrelation function C(k) takes on two values C(0) = 1 and C(k) = -1/15, for k ≠ 0

Combination generator

Example: Threshold generator

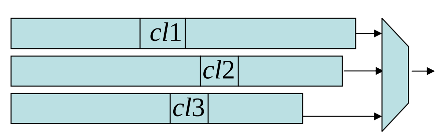


$t(x1, x2, x3) = 1$, if at least two of the inputs are equal to 1
0, otherwise

Clock Controlled generators

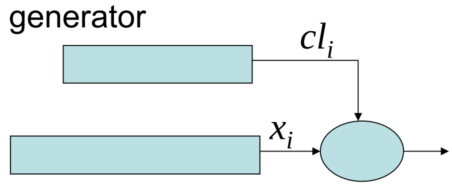
A clocking sequence is derived. The clocking sequence determines how the LFSRs are shifted

Example: A5/1



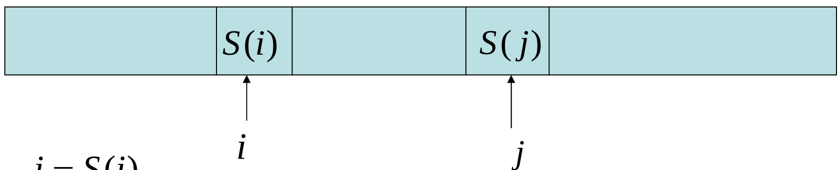
Clock bits are read. The LFSRs which are in majority,are shifted

Example: Shrinking generator
If the $cl_i = 0$,
then x_i is dropped



RC4

Register of 256 octets initialised using the key.
Counter i is set to zero. Then:



$$j = S(i)$$

$$S(i) \text{ are } S(j) \text{ swapped}$$

$$k = (j + S(j)) \bmod 256$$

$$\text{ouput} = S(k)$$

$$i = (i+1) \bmod 256$$

4.2 Block cipher confidentiality modes of operation

Block ciphers (in general) not good as such

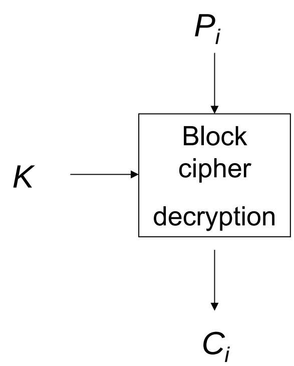
- AES modes of operations:
  - ELECTRONIC CODEBOOK MODE, ECB
  - CIPHER BLOCK CHAINING, CBC
  - CIPHER FEEDBACK , CFB
  - OUTPUT FEEDBACK , OFB
  - COUNTER MODE, CTR

standardised by NIST, Special Publication 800-38A ,see:
http://csrc.nist.gov/publications/nistpubs/index.html

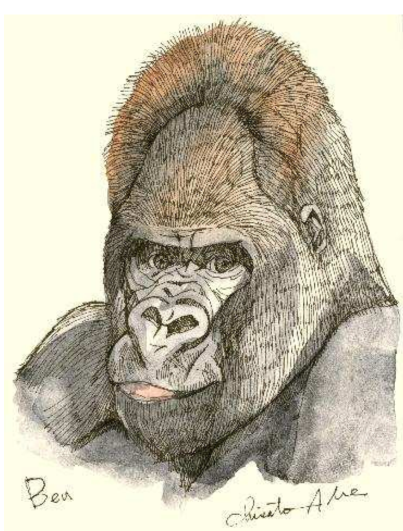
DES algorithm not good as such (small key size)

- Triple DES Special Publication 800-67

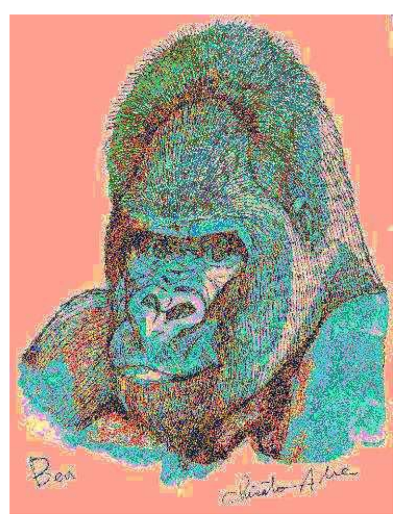
Electronic Code Book (ECB) Mode: Encryption



ECB encryption

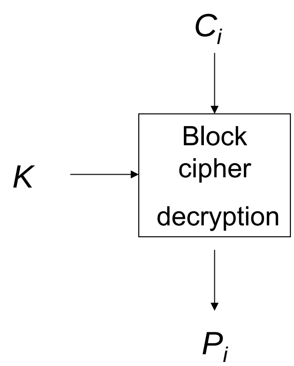


Plaintext



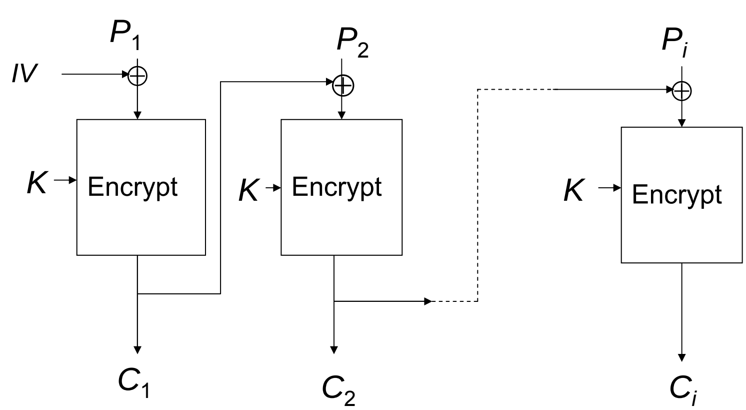
Ciphertext

Electronic Code Book Mode: Decryption



19

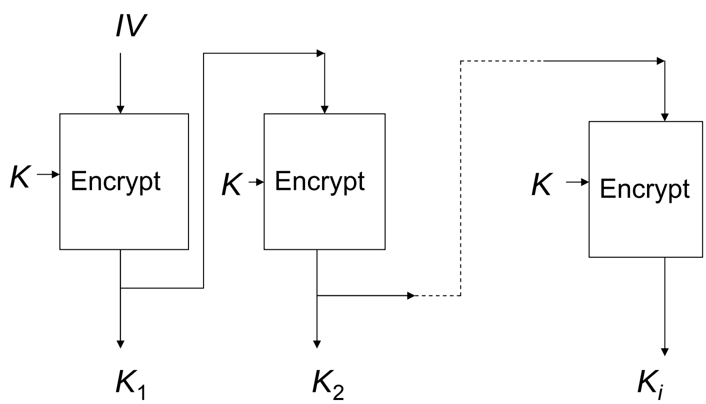
Cipher Block Chaining (CBC) Mode: Encryption



20

Output Feed Back (OFB) Mode

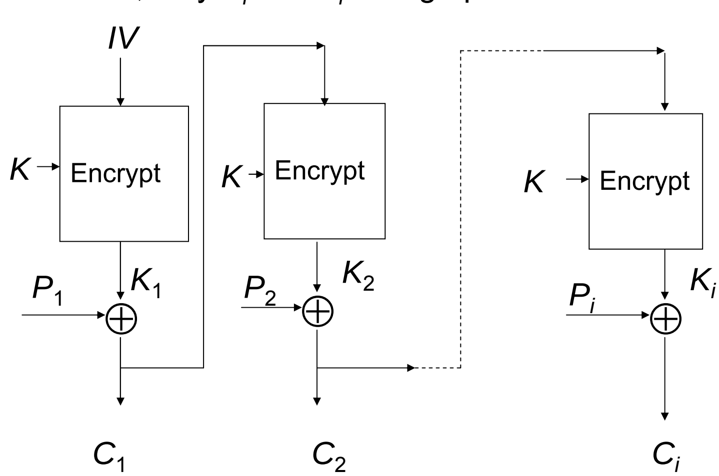
Synchronous Key Stream Generator:
Identical for encryption and decryption



21

Cipher Feed Back (CFB) Mode: Encryption

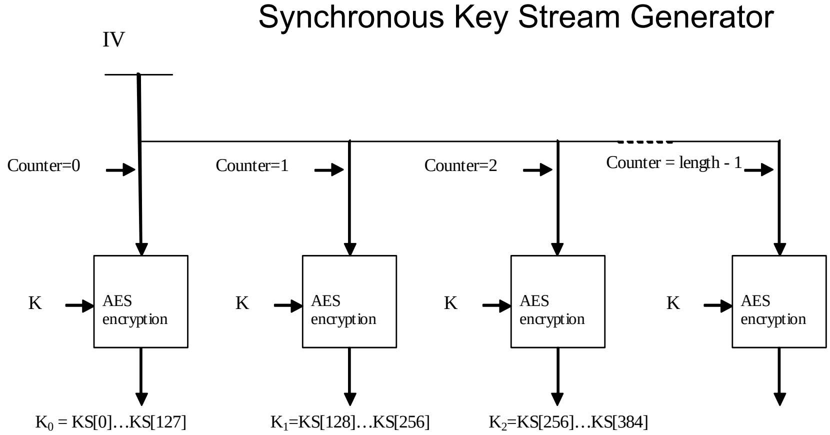
Self-Synchronising Stream Cipher: Decryption device is identical, only P_i and C_i change places



22

Counter Mode

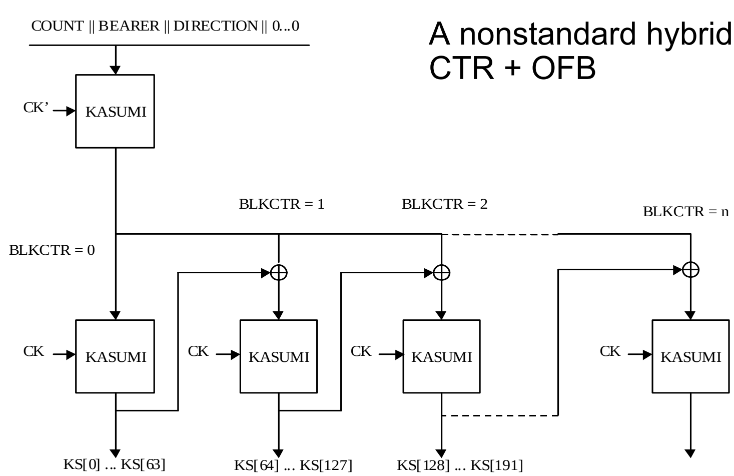
Synchronous Key Stream Generator



23

UMTS Encryption algorithm f8

A nonstandard hybrid mode:
CTR + OFB



CT[i] = PT[i] XOR KS[i]

Triple DES (TDEA)

DES algorithm not good as such (small key size)

Double DES with two different keys K_1 and K_2 not good either (security not more than single DES) due to the Meet-in-the-Middle Attack (see next slide):

Triple DES Special Publication 800-67, see

http://csrc.nist.gov/publications/nistpubs/index.html

Triple DES with two keys

$$C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$$

reduces to single DES, in case $K_1 = K_2$.

25

Meet in the Middle

Double DES with two different keys K_1 and K_2 not good: security is not more than single DES due to the Meet-in-the-Middle Attack. Such attack can be launched when the attacker has two known plaintext-ciphertext pairs (P,C) and (P',C'). For such pairs obtained using the secret keys K_1 and K_2 the attacker has

$C = E_{K_2}(E_{K_1}(P))$ and $C' = E_{K_2}(E_{K_1}(P'))$ or what is the same:

$D_{K_2}(C) = E_{K_1}(P)$ and $D_{K_2}(C') = E_{K_1}(P')$.

Now we make a table T with a complete listing of all possible pairs $K_2, D_{K_2}(C)$ as K_2 runs through all possible 2^{56} values. The table has 2^{56} rows with 120 bits on each row. We make one more column to this table, and fill it with K_1 values as follows: For each K_1 we compute the value $E_{K_1}(P)$ and search in the table T for a match $D_{K_2}(C) = E_{K_1}(P)$. For each K_2 we expect to find a (almost) unique K_1 such that such a match occurs. Now we go through all key pairs K_1 , K_2 suggested by table T, and test against the equation $D_{K_2}(C') = E_{K_1}(P')$ we have based on the second plaintext – ciphertext pair (P',C'). The solution is expected to be unique. The size of table T is 2^{56} (56 + 64 + ~56 bits) < 2^{64} bits, which is the memory requirement of this attack. The number of encryptions (decryptions) needed is about $4 \cdot 2^{56} = 2^{58}$.

26