- Shift cipher, Affine cipher, Substitution cipher, Vigenere cipher, Hill cipher, Permutation cipher, Playfair cipher

→ all block cipher

$x = x_1 x_2 \cdots$ (plaintext string)

$y = y_1 y_2 \cdots = e_k(x_1)\, e_k(x_2) \cdots$ (ciphertext string).

Same key K is used to encrypt successive plaintext elements

- Stream cipher (keystream is generated)

k (key)

$k_1 \quad k_2 \cdots$ (keystream)

$x_1 \quad x_2 \cdots$ (plaintext string)

$y_1 \quad y_2 \cdots = e_{k_1}(x_1)\, e_{k_2}(x_2) \cdots$ (ciphertext string)

- Block cipher is a special case of stream cipher when $k_i = k \; \forall i \geq 1$
- Synchronous steam cipher → keystream is constructed from the key, independent of the plaintext string

$g \rightarrow$ a keystream generator

$g(k) = k_1 k_2 \cdots$

Def: (Synchronous Stream cipher)

Six tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{L}, \mathcal{E}, \mathcal{D})$

- $\mathcal{P} \rightarrow$ plaintext span
- $\mathcal{C} \rightarrow$ ciphertext span
- $\mathcal{K} \rightarrow$ key span
- $\mathcal{L} \rightarrow$ keystream alphabet
- $g \rightarrow$ key stream generator

$g(k) = k_1 k_2 \cdots$

↳ generates an infinite string over $\mathcal{L}$ for $k \in \mathcal{K}$

- $\forall k_i \in \mathcal{L}, \exists\, e_{k_i} \in \mathcal{E}$ & $d_{k_i} \in \mathcal{D}$.

$e_{k_i} : \mathcal{P} \to \mathcal{C}$, $d_{k_i} : \mathcal{C} \to \mathcal{P}$ s.t.

$$d_{k_i}(e_{k_i}(x)) = x \quad \forall\ x \in \mathcal{P}$$

Example Vigenere cipher can be defined as a synchronous stream cipher.

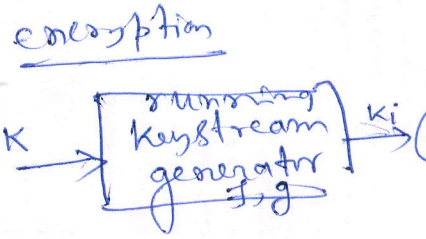
Example (Binary additive stream cipher)

$$\mathcal{P} = \mathcal{C} = \mathcal{K} = Z_2$$

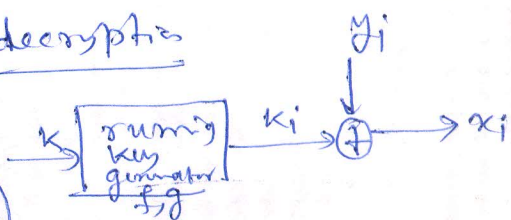
$$e_{k_i}(x_i) = (x_i + k_i) \bmod 2$$

$$d_{k_i}(y_i) = (y_i + k_i) \bmod 2$$

encryption



decryption



when $k_1 k_2 \ldots$ truely random → we get Vernam one time pad

Encryption (& decryption) can be efficiently implemented in hardware.

Example: Linear recurrence of degree m to generate key stream (Synchronous).

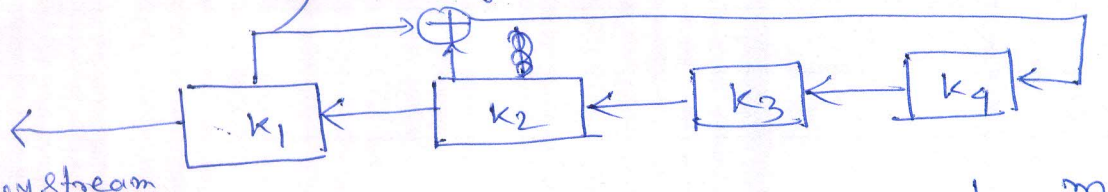
$$k_{i+m} = \sum_{j=0}^{m-1} c_j k_{i+j} \bmod 2, \quad i \geq 1, \quad c_0, c_1, \ldots, c_{m-1} \in Z_2, \quad c_0 = 1$$

Efficient Hardware implementation (LFSR) of key stream.

- Use a shift ~~registrar~~ register with m stages.
- $IV = (k_1, k_2 \ldots, k_m)$

1. k_1 would be tapped as the next keystream bit
2. $k_2, k_3 \ldots, k_m$ ~~Shifted~~ would be shifted one stage to the left
3. New value k_m would be computed to be $\sum_{j=0}^{m-1} c_j k_{j+1}$ (this is linear feedback)

- Concurrently perform the above three operations at each time unit.



- At any given point of time, the shift register contains m consecutive keyStream elements, $k_i, k_{i+1}, \cdots, k_{i+m-1}$.

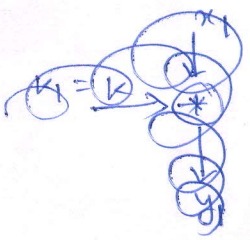
$m=4$
$k_{i+4} = (k_i + k_{i+1}) \bmod 2$.

- After one time unit, the shift register contains $k_{i+1}, k_{i+2}, \cdots, k_{i+m}$.

- The linear feedback is carried out by tapping certain stages of the register & computing Sum mod 2.

Non-Synchronous Stream cipher

→ each key stream element k_i depends on previous plaintext and/or ciphertext elements ($x_1, x_2, \cdots, x_{i-1}$ and/or $y_1, y_2, \cdots, y_{i-1}$) as well as the key K.



Example: (Autokey Cipher)

$P = C = K = \mathbb{Z}_{26}$

Key stream $k_1 = K$, $k_2 = x_{i-1}$ $\forall i \geq 2$, $K \in K$

For $0 \leq k_i \leq 25$, $e_{k_i}(x_i) = (x_i + k_i) \bmod 26$
$d_{k_i}(y_i) = (y_i - k_i) \bmod 26$.

Illustration (Autokey)

- key: $k = 8$
- Plaintext: rendezvous

17, 4, 13, 3, 4, 25, 21, 14, 20, 18

- keystream: $k_1 = k = 8$, $k_i = x_{i-1} \; \forall i \geq 2$

8, 17, 4, 13, 3, 4, 25, 21, 14, 20

- ciphertext:

25, 21, 17, 16, 7, 3, 20, 9, 8, 12

ZVRQHDUJIM

0	1	2	3	4	5	6	7
a	b	c	d	e	f	g	h
8	9	10	11	12	13	14	
i	j	k	l	m	n	o	
15	16	17	18	19	20		
p	q	r	s	t	u		
21	22	23	24	25			
v	w	x	y	z			

46
26
20

35
26

34 38
26
12

Decryption

$x_1 = 25 - 8 = 17 \mod 26$

$x_2 = 21 - 17 = 4 \mod 26$

⋮

- Autokey cipher is insecure as there are only 26 possible keys.

- Block ciphers → process plaintext in relatively large blocks. ($n \geq 64$ bits)
  - → Same fn is used to encrypt successive blocks.
  - → pure block ciphers are memoryless.

- Stream ciphers → process plaintext in blocks as small as a single bit.
  - → encryption fn may vary as plaintext is processed.
  - → has memory/~~state~~.
  - → encryption depends not only on key and plaintext, but also on current state.
  - → called state ciphers
  - ~~→ adding a small amount of memory~~

- Adding a small amount of memory to a block cipher (as in the CBC mode) results in a stream cipher with large blocks.

General model of a synchronous Stream cipher

$\sigma_{i+1} = f(\sigma_i, K)$

$k_i = g(\sigma_i, K)$

$y_i = h(k_i, x_i)$

σ_0 = initial state, may be determined from the key K.

f = next state fun^n.

g = keystream generator

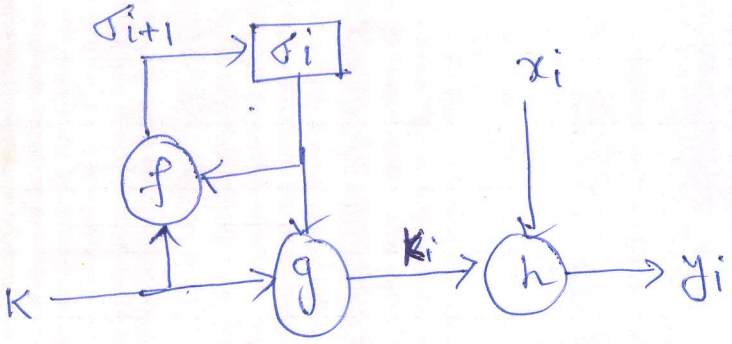
h = output fun^n

$k_1 k_2 \cdots$ = keystream.

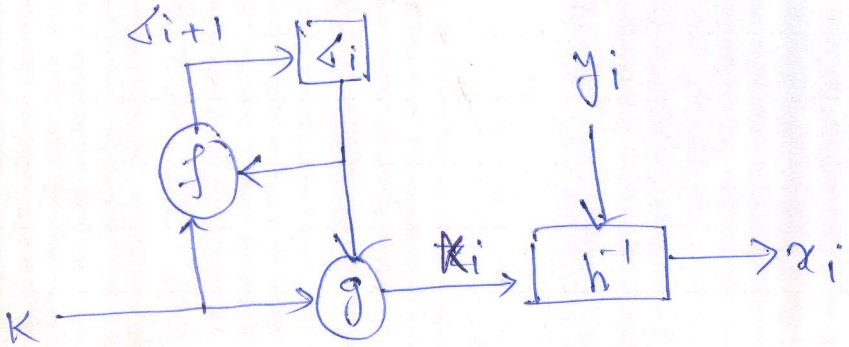
$x_1 x_2 \cdots$ = plaintext string

$y_1 y_2 \cdots$ = ciphertext string.

Encryption



Decryption

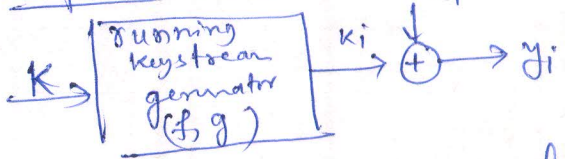


LFSR example

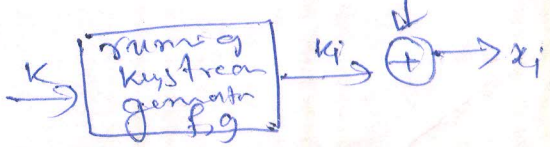
f → shifting of $k_2, k_3, \ldots, k_m$ to the left & recomputed k_m is (linear feedback)

g → outputs k_1 as next keystream bit

Binary additive stream cipher example (h = XOR)



- OFB mode of a block cipher is an example of a synchronous stream cipher.



Properties of Synchronous Stream ciphers

- both the Sender & ~~decryptor~~ receiver must be Synchronized to allow proper decryption.
  (use the same key & operating at the same state/position within the key)
- decryption fails if synchronization lost due to ciphertext digits during transmission.
  ↙ insertion or deletion of
  (Re-synchronization needed) — reinitialization, placing special markers at regular intervals in the ciphertext
- active attacks can be detected.
  (insertion, deletion or replay of ciphertext digits) ciphertext etc.)

- no error propagation.
  → modified ciphertext digits (not deleted) does not affect the decryption of other digits.
  → active adversary might be able to make changes to selected ciphertext digits, and know exactly what affect these changes have on the plaintext.
  (additional mechanism is needed to provide data origin authentication & data integrity guarantees)

↙ certain attack without getting detected.

General model of a Self-synchronizing Stream ciphers

→ keystream is generated as a fun. of the key and a fixed no. of previous ciphertext digits.

$$\sigma_i = (y_{i-t}, y_{i-t+1}, \cdots, y_{i-1})$$

$$k_i = g(\sigma_i, k)$$

$$y_i = h(k_i, x_i)$$

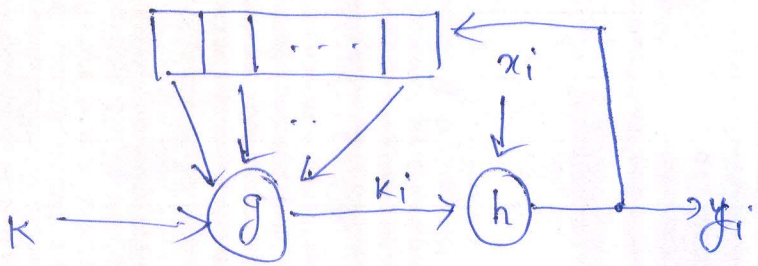
$\sigma_0 = (y_{-t}, y_{-t+1}, \cdots, y_{-1})$ is the initial state. (non-secret)

k = key

g = keystream generator

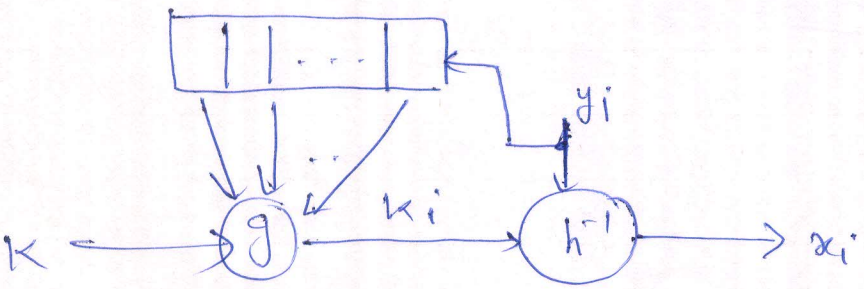
h = output fun.

Encryption



Example: most common presently used Self-sync. stream cipher is based on cipher feedback mode (CFB) block ciphers in 1-bit

Decryption



Properties of Self-synchronizing Stream ciphers

- Self-synchronization is possible if ciphertext digits are deleted or inserted as the decryption fn. depends only on a fixed no. of preceding ciphertext characters.
- Capable of re-establishing proper decryption automatically after loss of synchronization, with only a fixed no. of plaintext characters unrecovered.
- difficult to detect insertion, deletion or replay of ciphertext digits by an active adversary (additional mechanism is required to provide data origin authentication & data integrity).
- limited error propagation → Suppose a self-synchronization Stream cipher depends on t previous ciphertext digits. If a single ciphertext digit is altered (or even deleted or inserted) during transmission, then decryption of up to t subsequent ciphertext digits may be incorrect, after which correct decryption resumes.

→ Consequently, any modification of ciphertext digits by an active adversary causes several other ciphertext digits to be decrypted incorrectly, thereby improving the likelihood of being detected by the decryptor.

Modes of operation (Developed for DES).

- Can be used for any block cipher.

ECB → electronic codebook mode
CBC → cipher block chaining mode.
OFB → output feedback mode.
CFB → ciphertext feedback mode.

• ECB mode (naive use of a block cipher).

$x_1\ x_2 \ldots$ plaintext blocks.

$\downarrow\ \downarrow$

$y_1\ y_2 \ldots$ ciphertext blocks

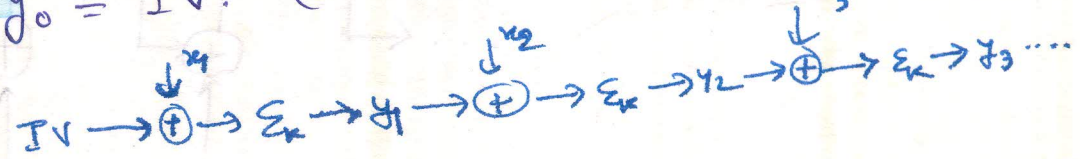
$y_i = E_k(x_i)$

- Weakness: identical plaintext blocks yields identical ciphertext blocks
  - Useless when plaintext blocks are chosen from a low entropy plaintext space
    e.g. plaintext consists entirely of 0's or entirely 1's.

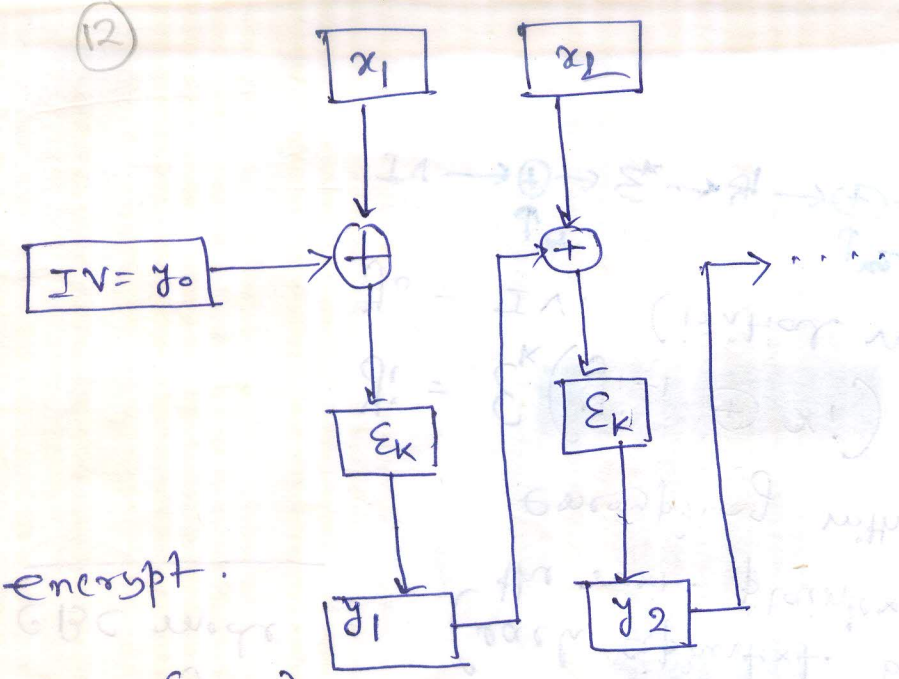
• CBC mode (each ciphertext block y_i is XORed with the next plaintext block x_{i+1} before encrypting with the key k).

$$y_i = E_k(y_{i-1} \oplus x_i),\ i \geq 1$$

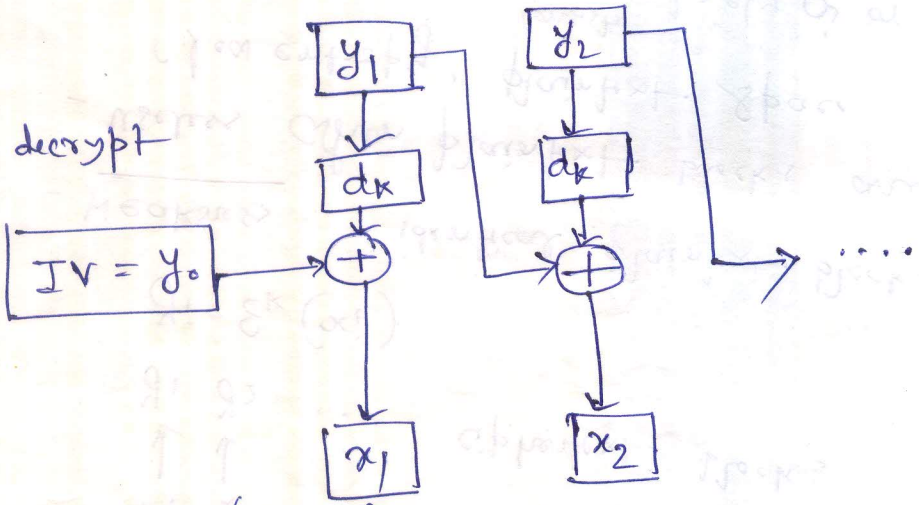
$y_0 = IV$. (initial vector).



(12)



$y_1 = E_K(y_0 \oplus x_1)$



$d_K(y_1) = (y_0 \oplus x_1) \oplus y_0 = x_1$
$\oplus y_0$

- Useful for authentication - to produce MAC (message authentication code).
  - if x_i is changed in CBC mode, then y_i & subsequent ciphertext blocks will be affected.
- plaintext || MAC. → provides the integrity, does not provide secrecy

OFB mode (operates as a synchronous stream cipher) → 1-bit OFB is synchronous stream cipher.

- Keystream is generated by repeatedly encrypting an initival vector IV.

Key → K.

Keystream → $k_1 k_2 k_3 \cdots$.

~~$k_0 = \mathcal{E}_K(IV)$~~

~~$k_i = \mathcal{E}_K$~~

$k_0 = IV$ (given).

$k_i = \mathcal{E}_K(k_{i-1})$, $i \geq 1$.

↓ keystream ind. of plaintext

plaintext → $x_1 x_2 \cdots$

ciphertext → $y_1 y_2 \cdots$

$y_i = x_i \oplus k_i$

decryption.

generate keystream $k_0 = IV$, $k_i = \mathcal{E}_K(k_{i-1})$, $i \geq 1$.

plaintext bits → $x_i = y_i \oplus k_i$

- The encryption fun. $\mathcal{E}_K$ is used for both encryption & decryption.

CFB mode (operates as a asynchronous stream cipher). → 1-bit CFB → asynchronous stream cipher

- Keystream is produced by encrypting the previous ciphertext block.

Key → K

Keystream → $k_1 k_2 \cdots$

plaintext → $x_1 x_2 \cdots$

ciphertext → $y_1 y_2 \cdots$

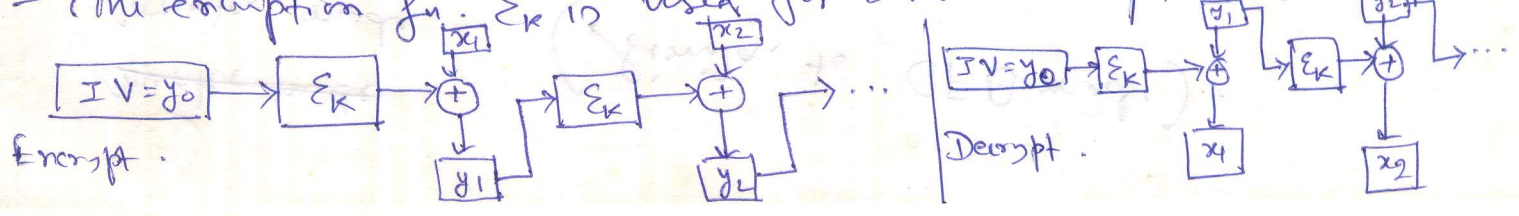
$y_i = x_i \oplus k_i$

$y_0 = IV$ (given)

$k_i = \mathcal{E}_K(y_{i-1})$, $i \geq 1$.

↓ keystream ind. of plaintext

- The encryption fun. $\mathcal{E}_K$ is used for both encryption & decryption.



Counter mode (similar to OFB mode).

- Keystream generation is different.
- plaintext block length x.
- Construct a seq. of bitstrings of length m each:

  $T_1, T_2 \ldots$

  using a counter ctr.

$$T_i = ctr - i + 1 \bmod 2^m, \quad i \geq 1.$$

plaintext: $x_1 x_2 \ldots$

ciphertext: $y_1 y_2 \ldots$

$$y_i = x_i \oplus e_K(T_i), \quad i \geq 1.$$

↓ keystream ind. of plaintext.

- unlike OFB $(K_i = E_K(K_{i-1}))$, $e_K(T_i)$ can be computed independently of any other keystream element — prior plaintext

→ $e_K(T_i)$ does not depend of plaintext or prior $e_K(T_{i-1})$

- permits very efficient implementation in software or hardware by exploiting opportunities for parallelism.

mode

↳ counter mode + CBC mode [counter with cipher block chaining mode]
(privacy) (authentication)