# Stream Ciphers

**Dr. E.SURESH BABU**

**Assistant Professor**

**Computer Science and Engineering Department**

**National Institute of Technology, Warangal**

**Warangal**

# Outline

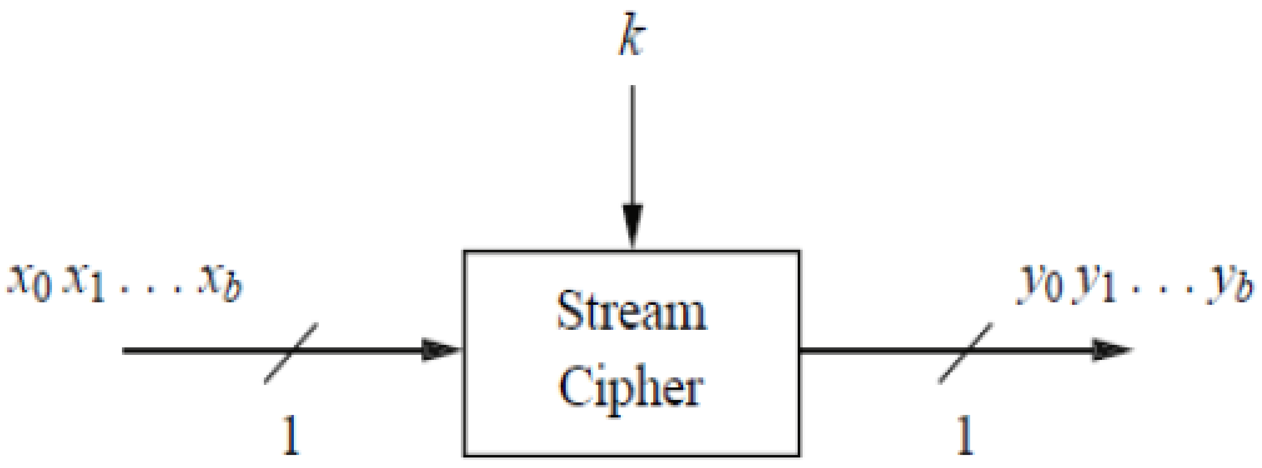
- ❖ **Symmetric Algorithm Works**
- ❖ **Stream Cipher**
- ❖ **Pseudorandom Number Generation**
  - ✓ **Types of Random Number**
- ❖ **Types of Stream Ciphers**
  - ✓ **Asynchronous or Self Synchronizing Stream Cipher**
  - ✓ **Synchronous Stream Cipher**
    - ➢ **Linear Feedback Shift Register**

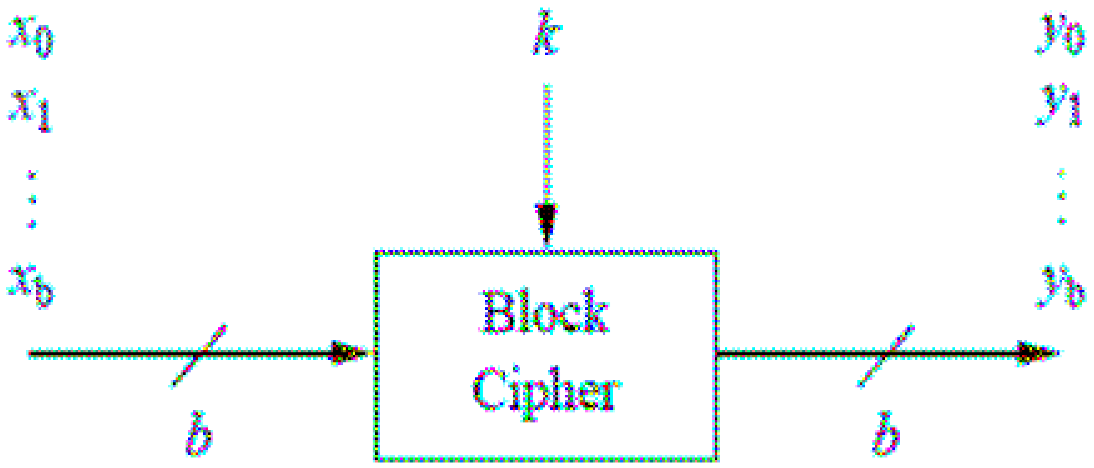
# Symmetric Algorithm Works

# Symmetric Algorithm Works

- **Symmetric Algorithms** can be divided into **two categories.**
  - ✓ Some **Symmetric Algorithms** operate on **SINGLE BIT or BYTE (STREAM CIPHER)**
  - ✓ Some Other **Symmetric Algorithms** operate on **GROUPS OF BITS (BLOCK CIPHER)**

# Stream Cipher Vs Block Cipher



# Difference

| Metrics | Stream ciphers | Block Ciphers |
|---|---|---|
| **Encryption** | Groups of characters (in blocks) | Individual characters (usually bits) |
| **Data Buffering** | None of limited required | More space required |
| **Speed** | Faster | Slower |
| **Hardware Circuitry** | Simpler | More complex |
| **Error propagation** | Limited – good for noisy channels | Propagates- good for assuring message integrity |
| **Software Implementation** | Not amenable | More efficient |

# Stream Cipher

# Introduction

- Some **Symmetric Algorithms** operate on the **plaintext** a **SINGLE BIT or BYTE** at a time;
  - ✓ These are called **Stream Algorithms** or **Stream Ciphers**.

# Introduction

- We will now **focus on ciphers** that are **designed explicitly** to work as **Stream Ciphers.**
  - ✓ As you already know, a **typical Stream Cipher** encrypt **single characters of plaintext one by one** with **time varying transformations.**

# Advantages of Stream Ciphers

- As the **stream ciphers** encrypt individual digits
  - ✓ It takes **less buffer memory**
  - ✓ **Less complex hardware circuitry** and
  - ✓ **Comparatively faster** than **Block ciphers.**

# Applications of a Stream Cipher

❖ These are the some areas where **stream ciphers can be useful**

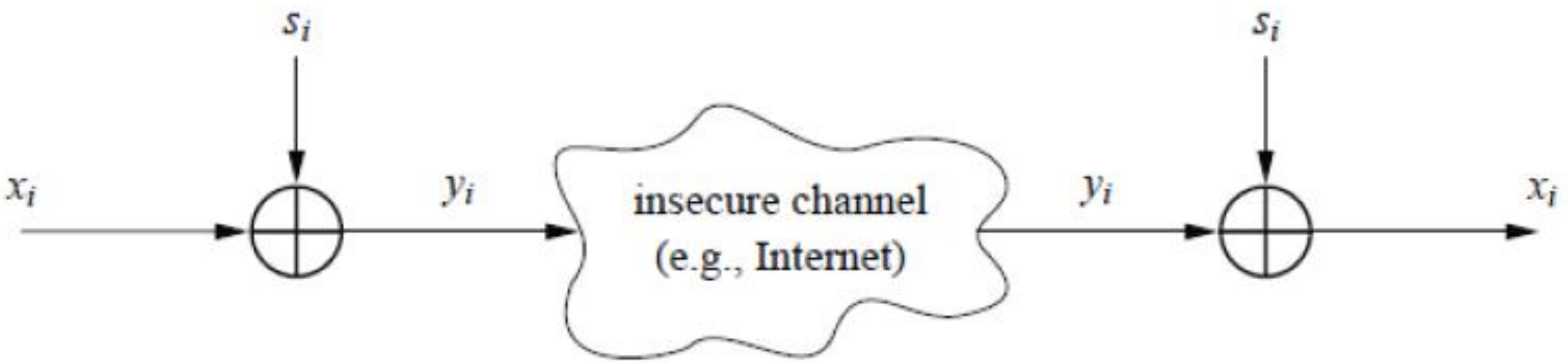
- ✓ It can be used in **RFID tags and Smart cards.**
- ✓ it are also desirable where **zero error propagation** is required like **radio communication**
- ✓ it are also desirable to use in **GSM communication.**
- ✓ it is particularly appropriate for **audio and video streaming.**
- ✓ it is also frequently used for **browser – web-server links**.

# One Time Pad (Recap)

# Vernam Cipher or OTP.

- Recall the **unbreakable Vernam cipher.**
- A **Vernam cipher** over the **binary alphabet** is defined by:

$$c_i = m_i \oplus k_i, \ \text{for } i = 1, 2, 3, \ldots$$



# Drawbacks of OTP.

- For almost all applications the **OTP is impractical**
  - since the **key** must be **as long as the message!**
    - Imagine you have to **encrypt a 1GByte email attachment.**
  - Does not guarantee **integrity.**
    - One-time pad only guarantees **confidentiality**
  - **Insecure** if keys are reused.
    - Attacker can obtain **XOR of plaintexts**

# Design Goals of Stream Cipher

# Design Goal of Stream Cipher

- Design goal is to **efficiently produce random-looking sequences** possibly from **truly random sequences.**
  - ✓ Replace **“random”** with **“pseudo-random”**
  - ✓ Encrypt with **pseudo-random number generator (PRNG)**
  - ✓ PRNG takes a **short, truly random secret key** and expands it into a **long “random-looking” sequence**

# Why Pseudorandom Number Generation is Required

# Why Pseudorandom Number Generation is Required

- Secure communications in computer networks would simply be impossible without **high quality random** and **pseudorandom number generation.** Here are some of the reasons:

# Some of the Reasons:

- The **session keys that a KDC** must generate on the fly are nothing but a sequence of **randomly generated bytes.**
- The **Nonce** that are exchanged during handshaking between a **host and a KDC** are also **random numbers.**
- **Random numbers** are also needed for the **RSA public-key encryption algorithm.**
  - ✓ RSA needs are **prime numbers.**

# Some of the Reasons:

- Random numbers are also used to serve as **salts** in **password hashing schemes.**
- True Random Numbers are used to serve as **one-time keys.**

# Types of Random Number

# Types of Random Number

- **Cryptographic applications** typically make use of algorithmic techniques for **generation of random number**

    1. **True Random Number**
    2. **Pseudorandom Number**

# True Random Number

# When the Random Numbers are Truly Random

- To be considered **truly random**, a **sequence of numbers** must exhibit the following two properties:

1. **Uniform Distribution:** This means that **all the numbers** in a designated range must **occur equally often.**
2. **Independence:** This means that if we **know** some or all the number up to a **certain point in a random sequence**, we should **not** be able to **predict the next one** (or any of the future ones).

# True Random Number

- Truly random numbers can only be generated by **physical phenomena** such as thermal noise, cards, dice etc.
- **Modern computers** try to approximate **truly random numbers through** a variety of approaches that we will address in next section

# Pseudorandom Number

# Pseudorandom Number

- **Algorithmically** generated random numbers are called **pseudorandom numbers.**
- A pseudorandom sequence of numbers is cryptographically secure if it is **difficult for an attacker to predict the next number** from the numbers already in his/her possession.

# Types of PRNGs

- we look at **two types of algorithms** for **PRNGs.**

1. **Linear Congruential Generators**
2. **Blum Blum Shub Generator**

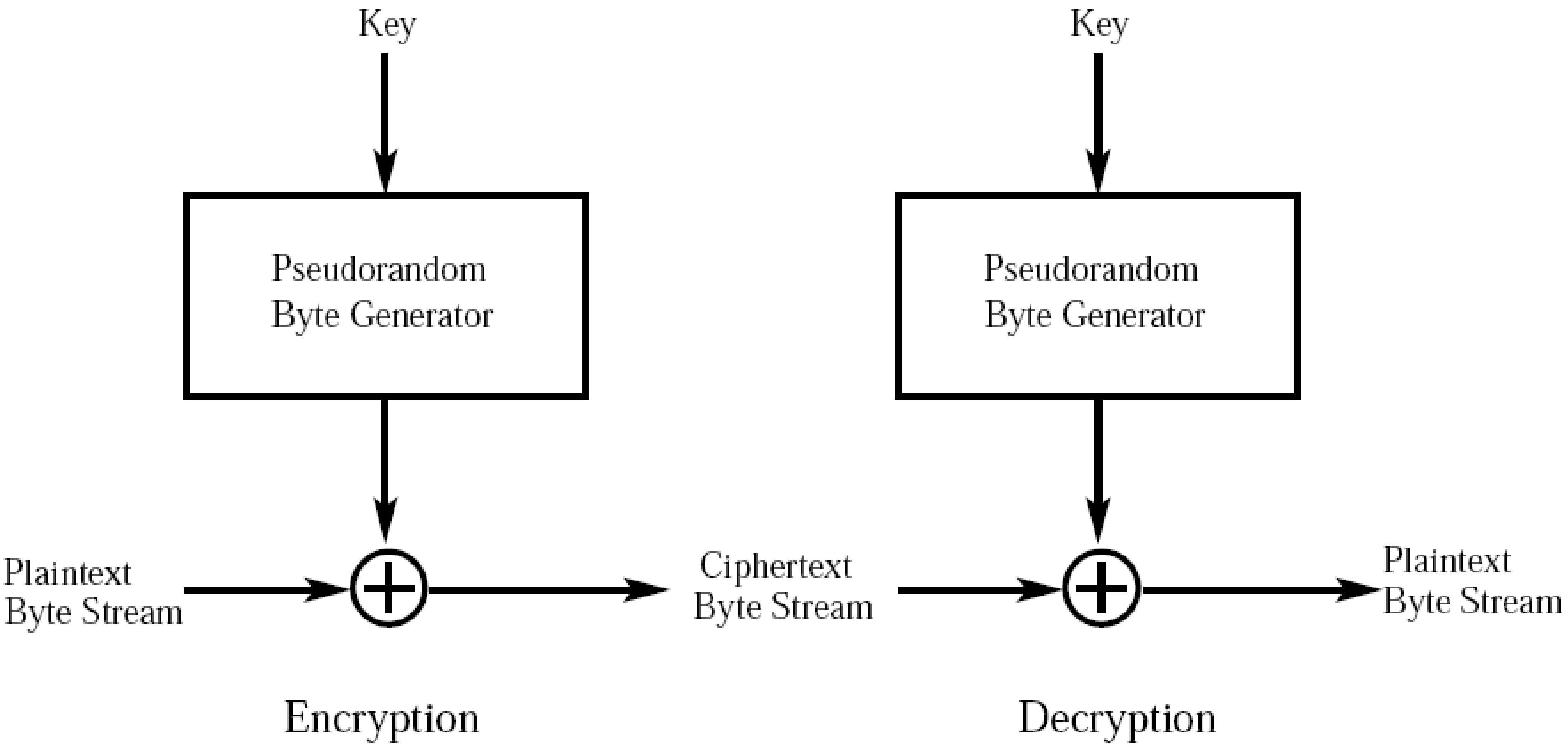
# True Stream Cipher

# True Stream Cipher

- The main processing step in a **True stream cipher** is the **generation of a stream of pseudorandom bytes** that depend on the **encryption key.**

- For each **different** encryption key will result in a **different stream** of pseudorandom bytes.

# Operation of a True Stream Cipher



# Operation of a True Stream Cipher

- Encryption itself is as simple as it can be.
  - ✓ Just **XOR** the **byte from the pseudorandom stream** with the **plaintext byte** to get the encrypted byte.
- You generate the **same pseudorandom byte stream** for decryption.
  - ✓ The decryption itself consists of **XORing the received byte** with the **pseudorandom byte.**

# Security of a True Stream Cipher

- For a **stream cipher** to be **secure**, the **pseudorandom sequence of bytes** should have as **long a period as possible.**
  - ✓ The **longer the random sequence** it is more difficult it is to **break the cipher.**
- To **Resist the brute-force attacks**, the encryption key should be as **long as possible.**
  - ✓ A desirable key length is **128 bits.**

# Types of Stream Ciphers

# Types of Stream Ciphers

- **Stream ciphers** are classified into **two classes**

  1. **Synchronous Stream Cipher**

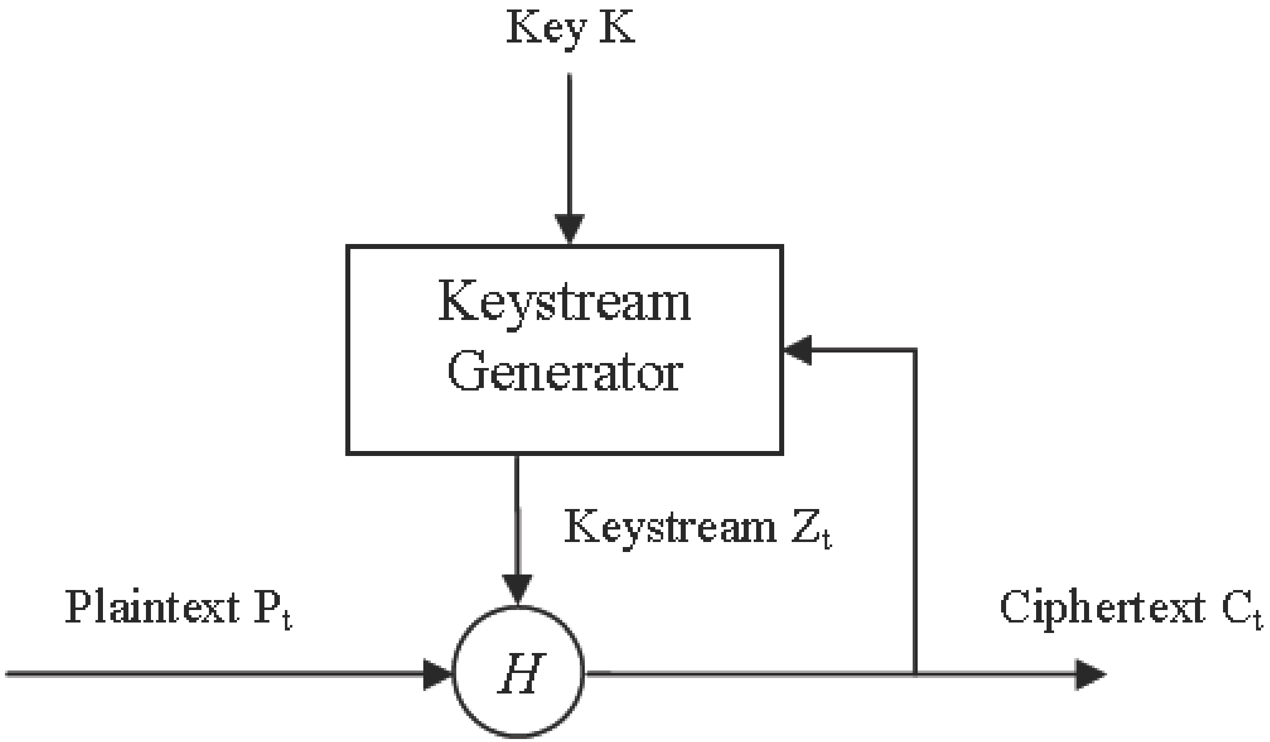
  2. **Asynchronous or Self Synchronizing Stream Cipher**

# Asynchronous or Self Synchronizing Stream Cipher

# Asynchronous Stream Cipher

- **Asynchronous stream cipher** work on the basis of **use of ciphertext in keystream generation.**
- The generated keystream is **dependent on the key** as well as **previous ciphertext digits**
- The **outputted ciphertext bits are inputted** in keystream generator for **state update of the cipher.**

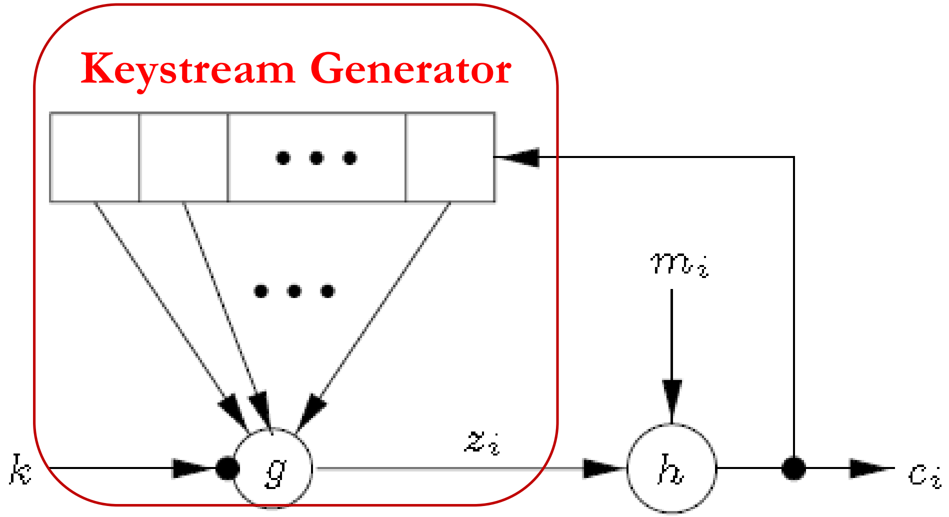
# Asynchronous Stream Cipher



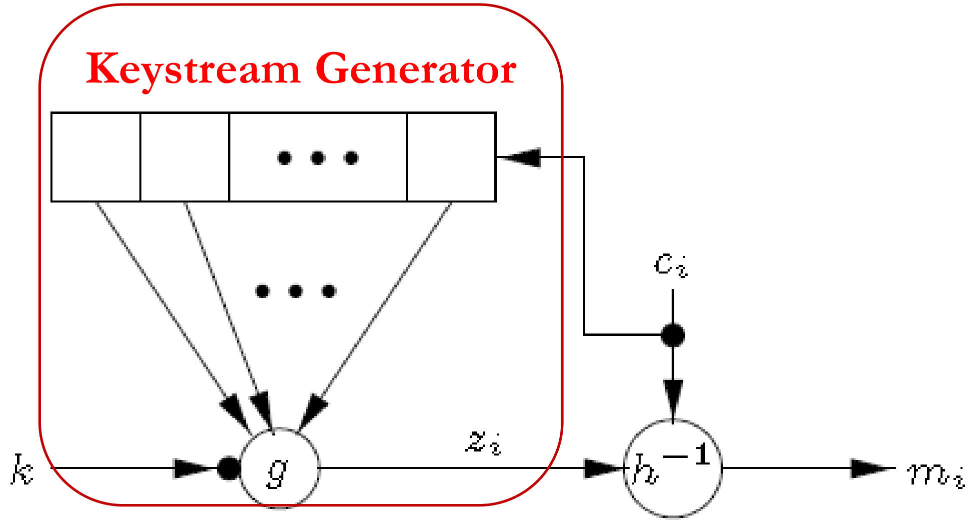
# Asynchronous Stream Cipher (State Update)

❖ The **outputted ciphertext bits are inputted** in **keystream generator** for **state update of the cipher.**

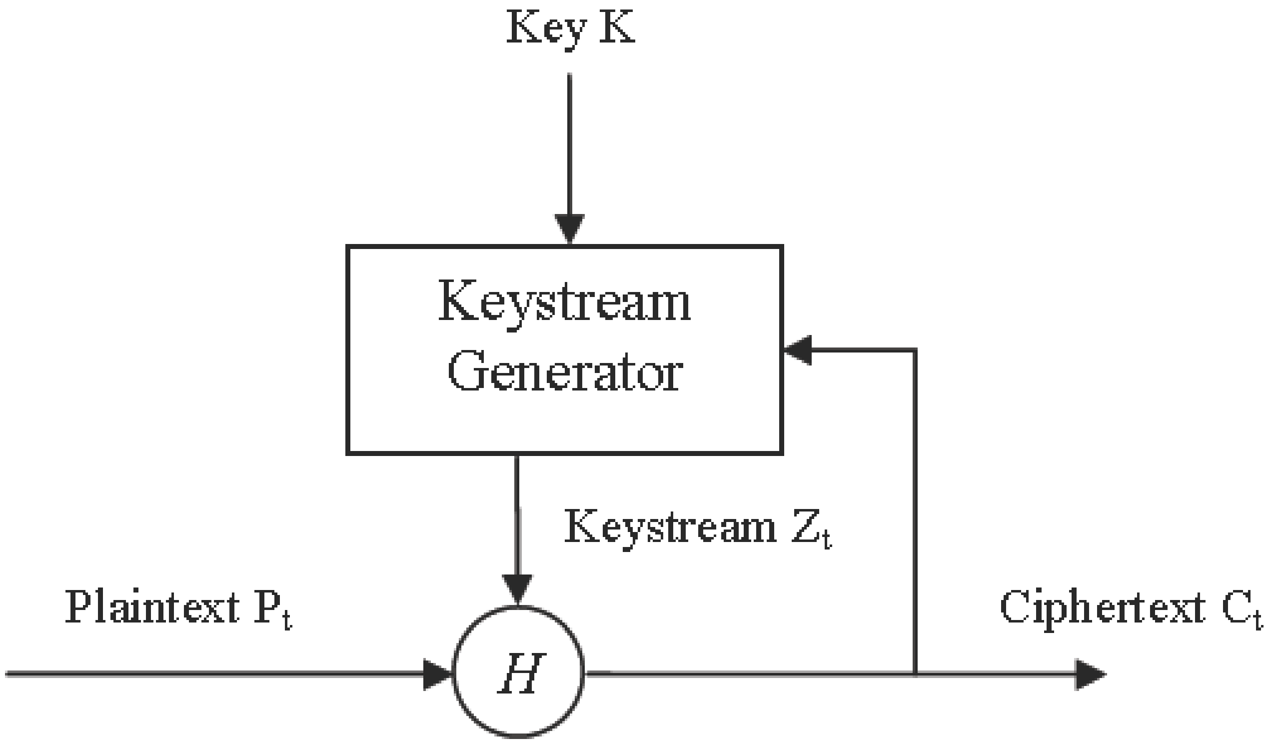
(i) Encryption




(ii) Decryption



# Asynchronous Stream Cipher



# Asynchronous Stream Cipher

- **Asynchronous Stream cipher equations** represented below the **different functions.**
  - **State Update Function:** $S_{t+1} = U(S_t, K, C_t)$
  - **Keystream Generation Function:** $Z_t = G(S_t, K)$
  - **Output Function Or Encryption Function** : $C_t = H(P_t, Z_t)$

# Observation

- All the **new states of the cipher** are **dependent** on the **previous of the cipher**
- In other words **encrypted or ciphered bits** generated from the **previous state.**

# Limitations

- The **initial state** is derived from the **key and IV bits** and in the **majority of ciphers IV** is kept public
- **Attackers** are well aware of the **some bits** that have been used in the **encryption.**
- Hence this type of ciphers is very **vulnerable to cryptanalytic attacks**.

# Limitations

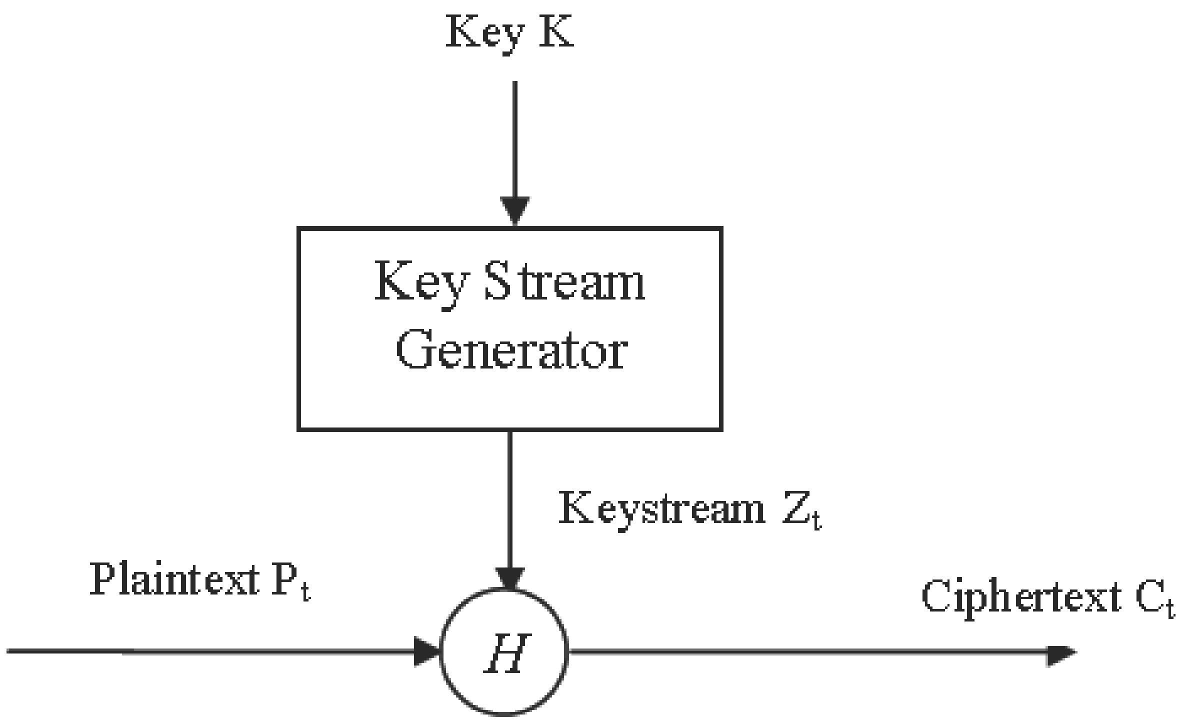
- The **effect of one single bit** is propagated to **n number of bits** and hence a **single error will propagate to n number of other bits.**
- These **weaknesses** make **Self Synchronizing Stream Ciphers** very less attractive and rarely used.

# Synchronizing Stream Cipher

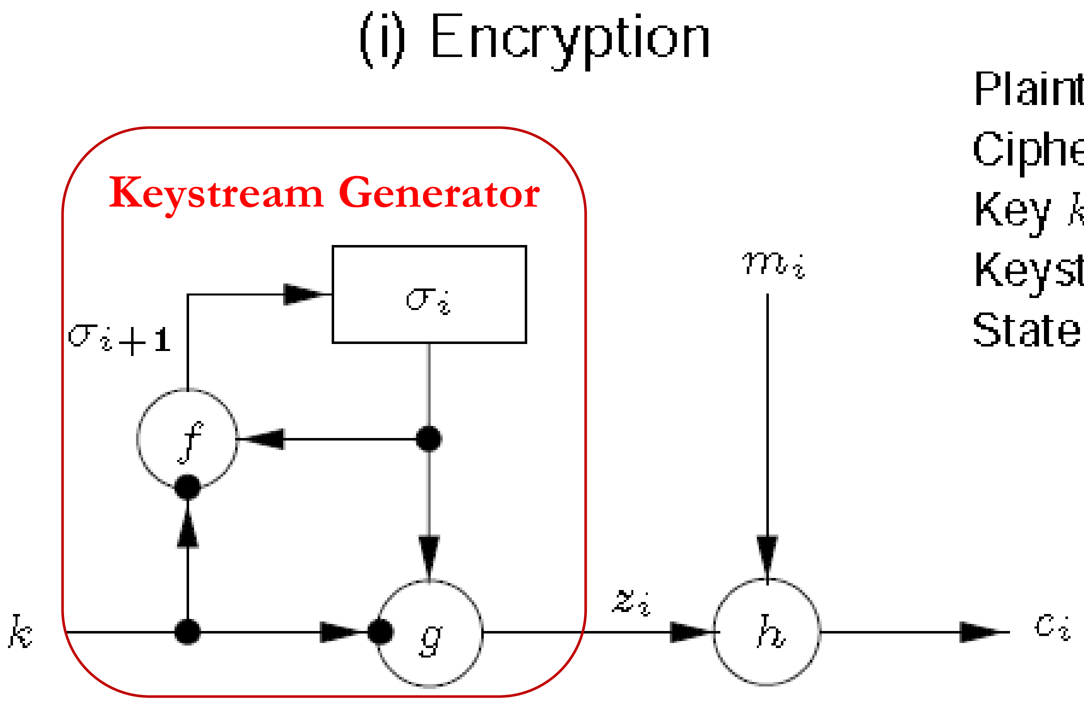
# Synchronous Stream Cipher

- In Synchronous stream cipher,
  - ✓ The **keystream is dependent only** on the **key**
  - ✓ There is **no relation** with the **previous ciphertext digits.**
  - ✓ The **secret key** and **state of the keystream generator** is used **only for the keystream generation**.

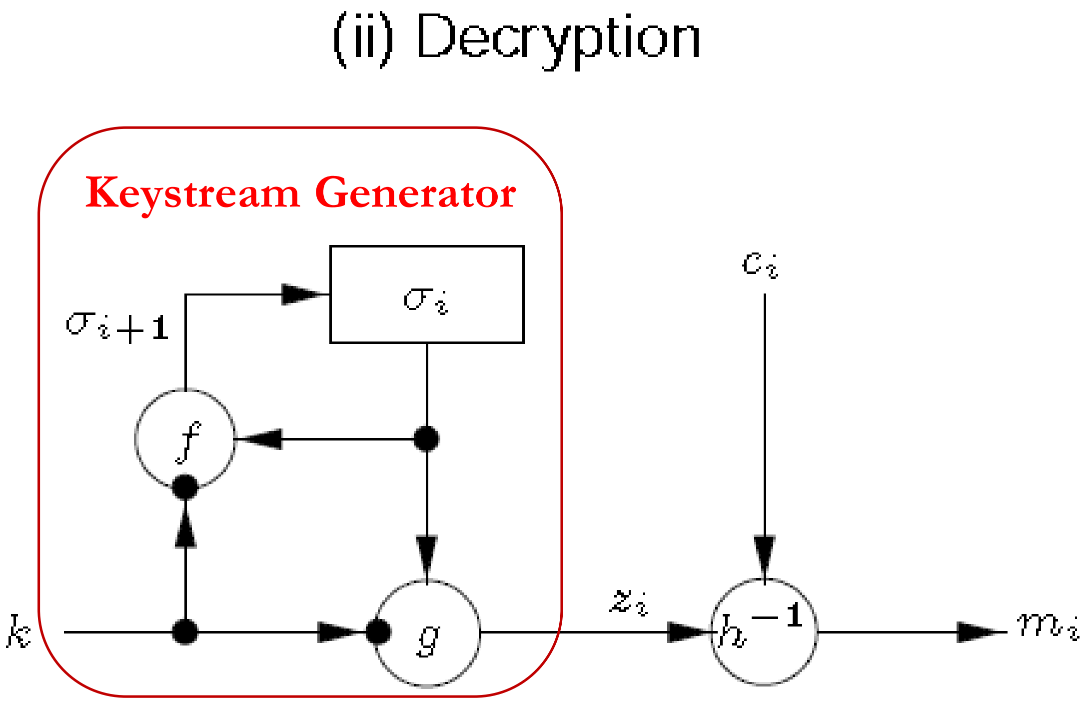
# Synchronous Stream Cipher



# Synchronous Stream Cipher with Key Stream Generator




Plaintext $m_i$
Ciphertext $c_i$
Key $k$
Keystream $z_i$
State $\sigma_i$



# Synchronous Stream Cipher

- **Synchronous Stream cipher equations** represented below the **different functions.**
  - **State Update Function:** $S_{t+1} = U(S_t, K)$
  - **Keystream Generation Function:** $Z_t = G(S_t, K)$
  - **Output Function Or Encryption Function** : $C_t = H(P_t, Z_t)$

# Observation

- The **keystream generation** is **independent of the previous ciphertext** generated
  - ✓ if an **error occurs at on bit**, it will **affect only one corresponding bit** at decryption stage.

# General Structure of Synchronous Stream Cipher

- **Any synchronous stream cipher** works in **two phases:**

  1. **Key Initialization or Key Setup Phase**

  2. **Key Stream Generation Phase**

# Key Initialization or Key Setup Phase

- In the **key initialization phase,**
  - ✓ A **secret key K and initialization vector IV** are used to generate the **initial state of the cipher.**

$$S_0 = \text{finit}\ (K, IV)$$

# Observation...

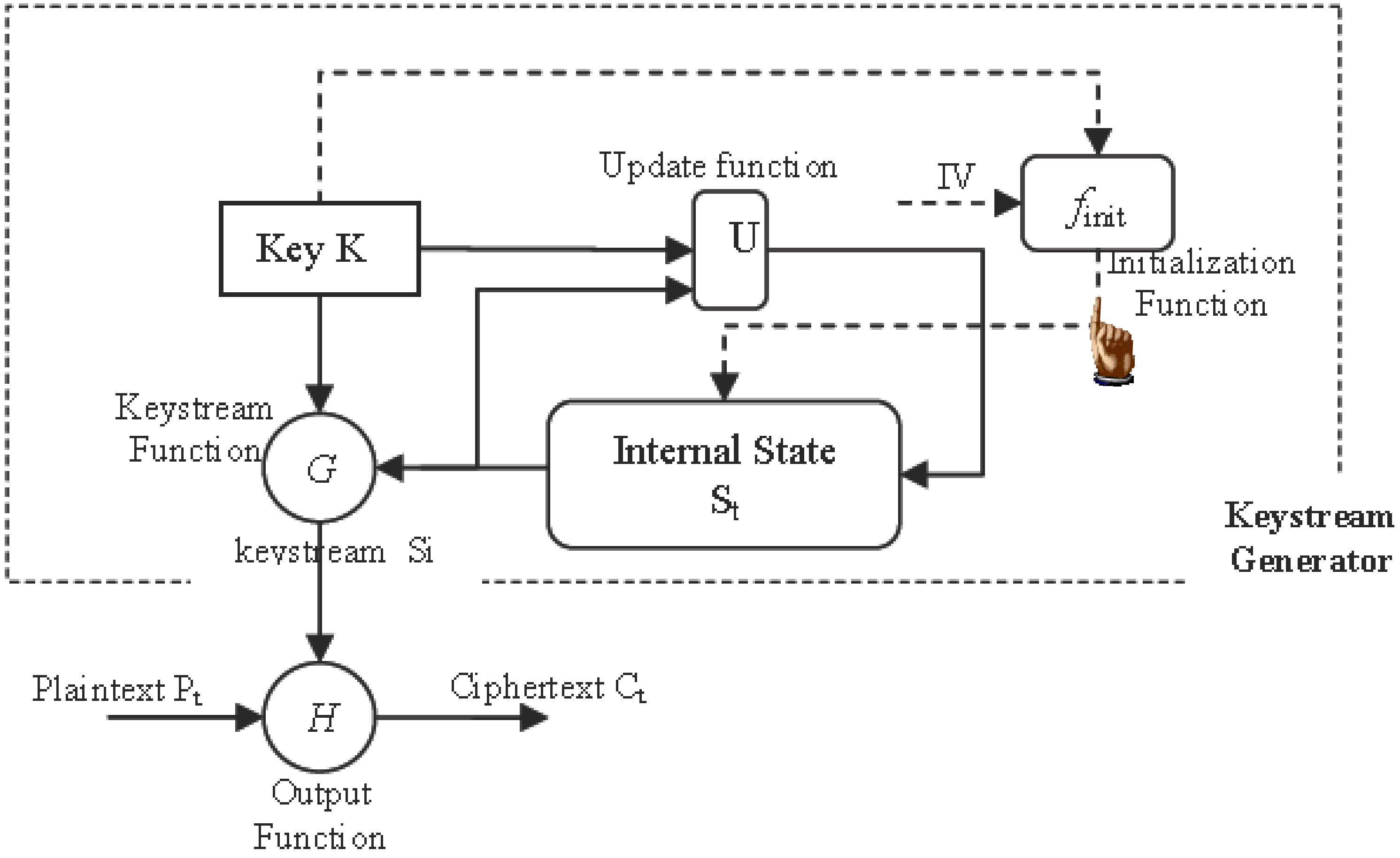
- After the **initial state of the cipher is generated** or **key setup phase completed**,
  - ✓ The **IV is not used** for **key generation.**

# Key Stream Generation Phase

- **Keystream** is generated with the **use of secret key and internal state** for the keystream generation using **keystream function G.**
- The **Output function H** is used with **keystream $Z_t$** and **plaintext $P_t$** to generate **ciphertext $C_t$.**

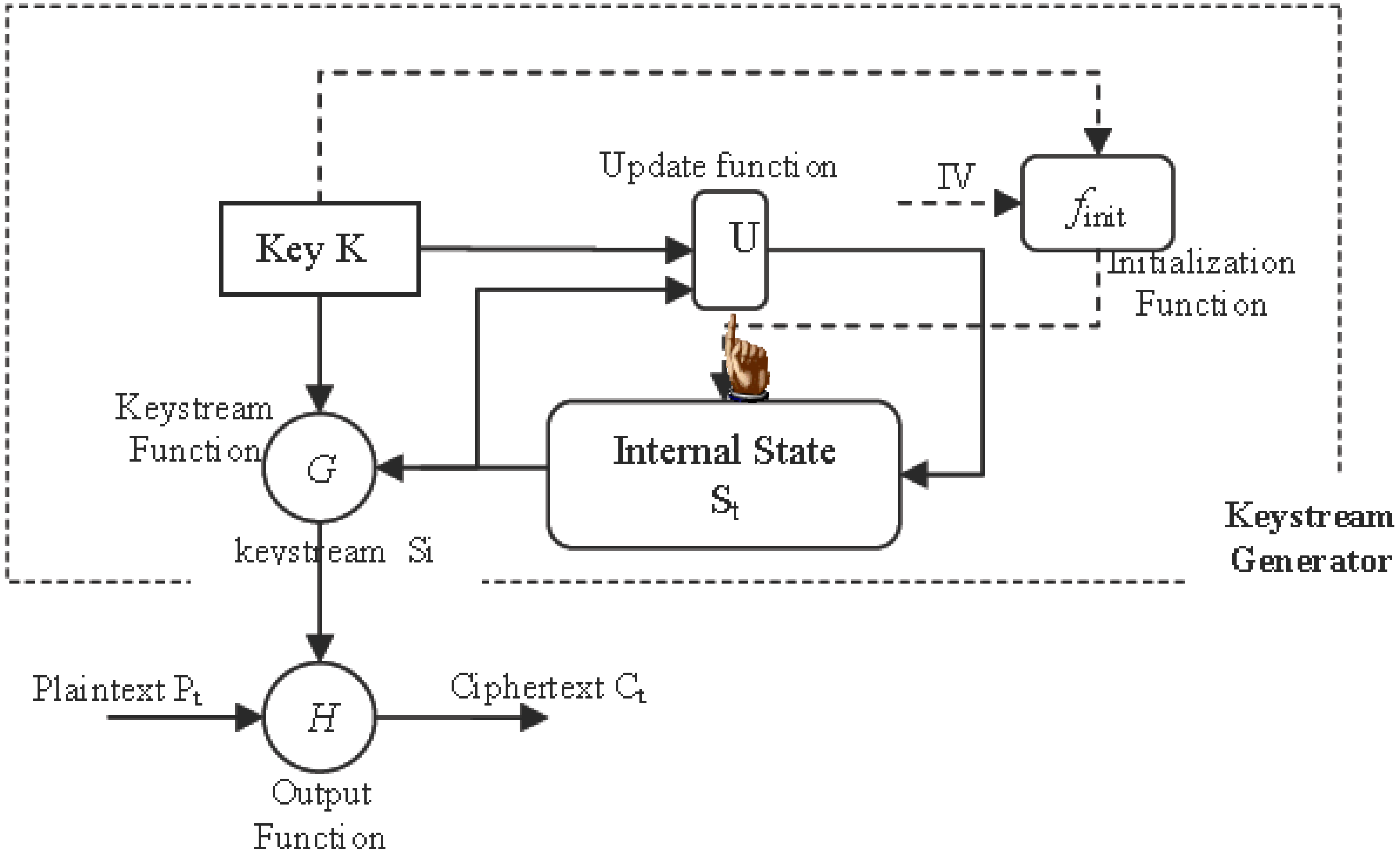
# Key Stream Generation Phase

- General Structure of the **synchronous stream cipher**



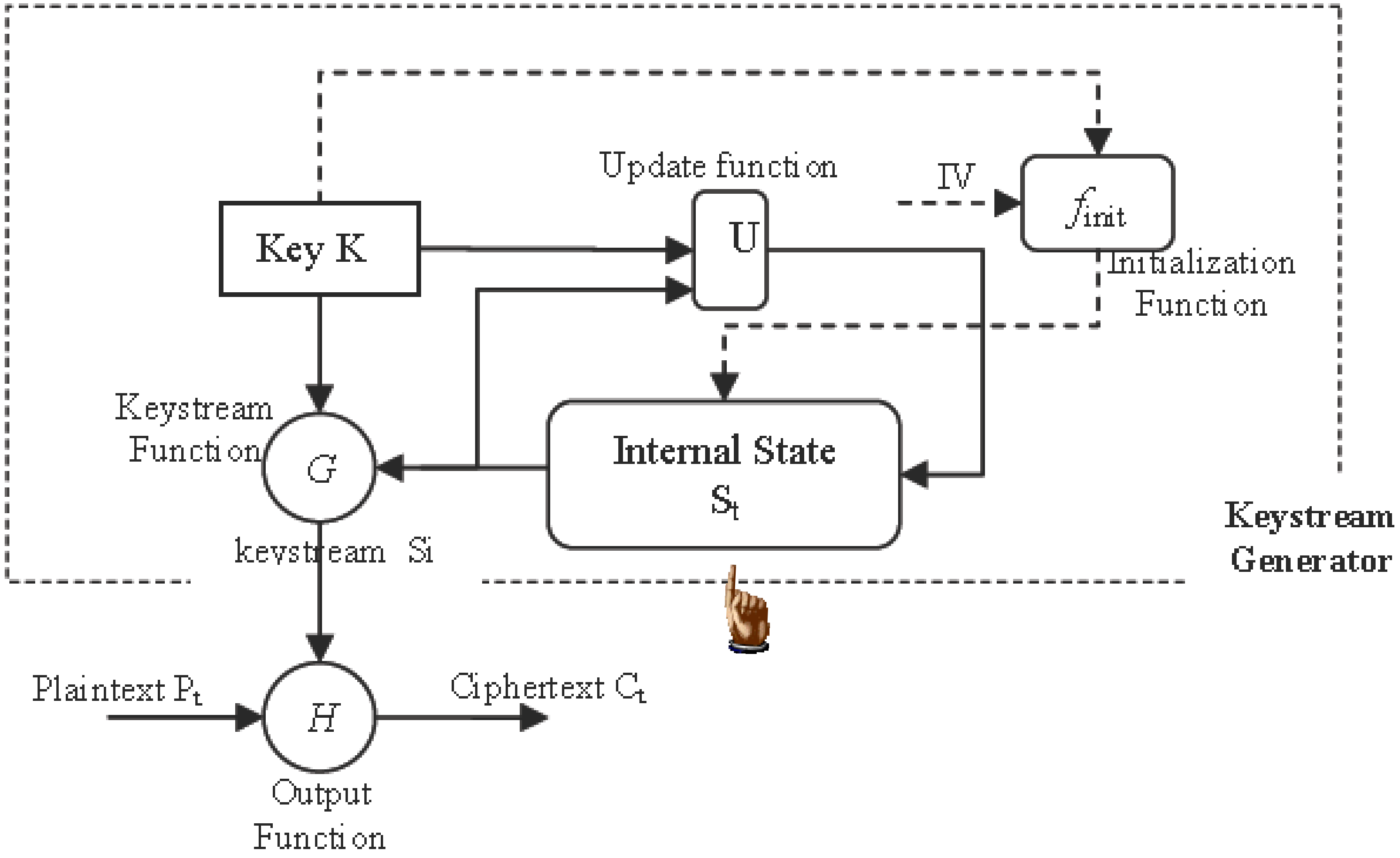
# Key Stream Generation Phase

❖ General Structure of the **synchronous stream cipher**



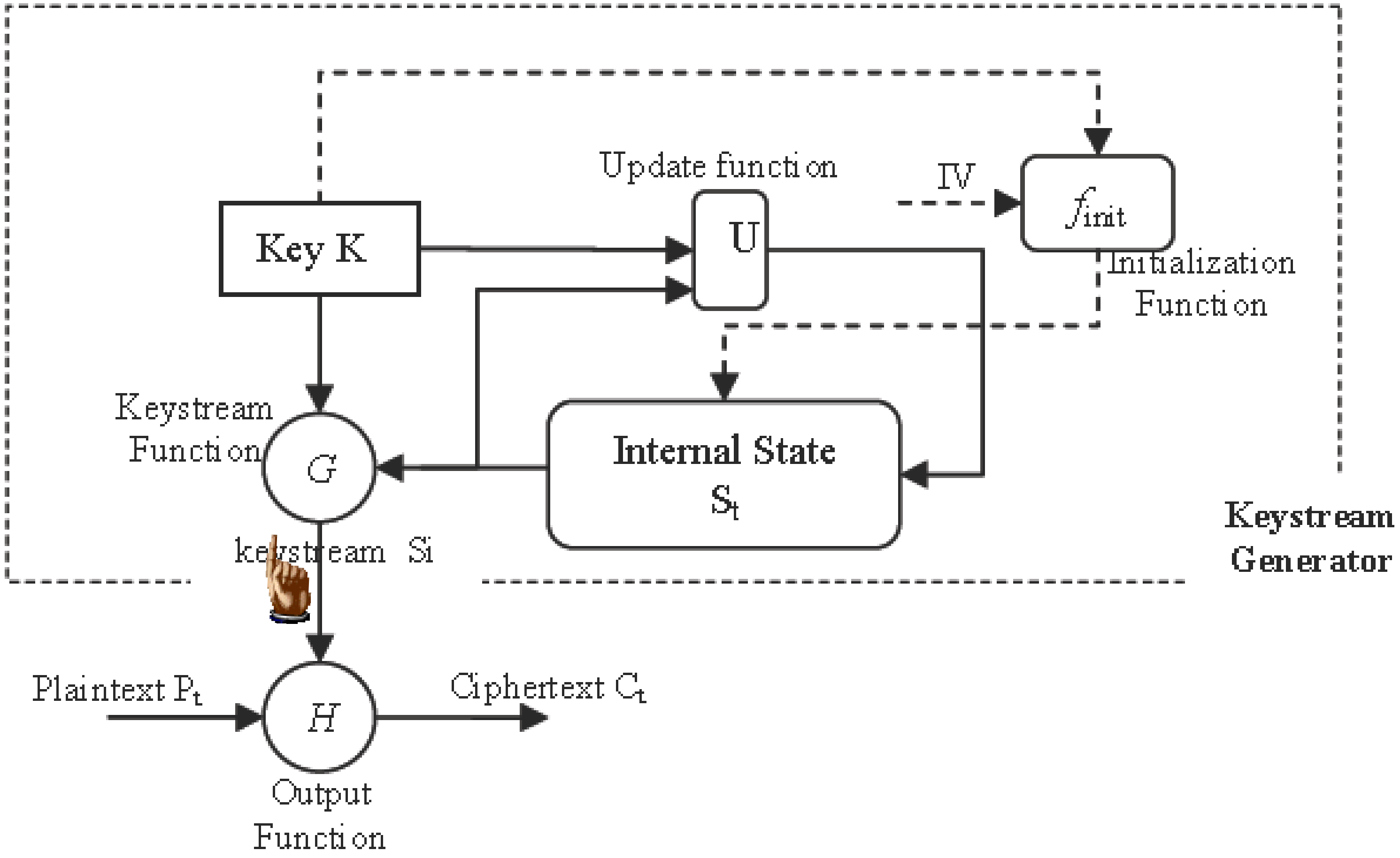
# Key Stream Generation Phase

❖ General Structure of the **synchronous stream cipher**



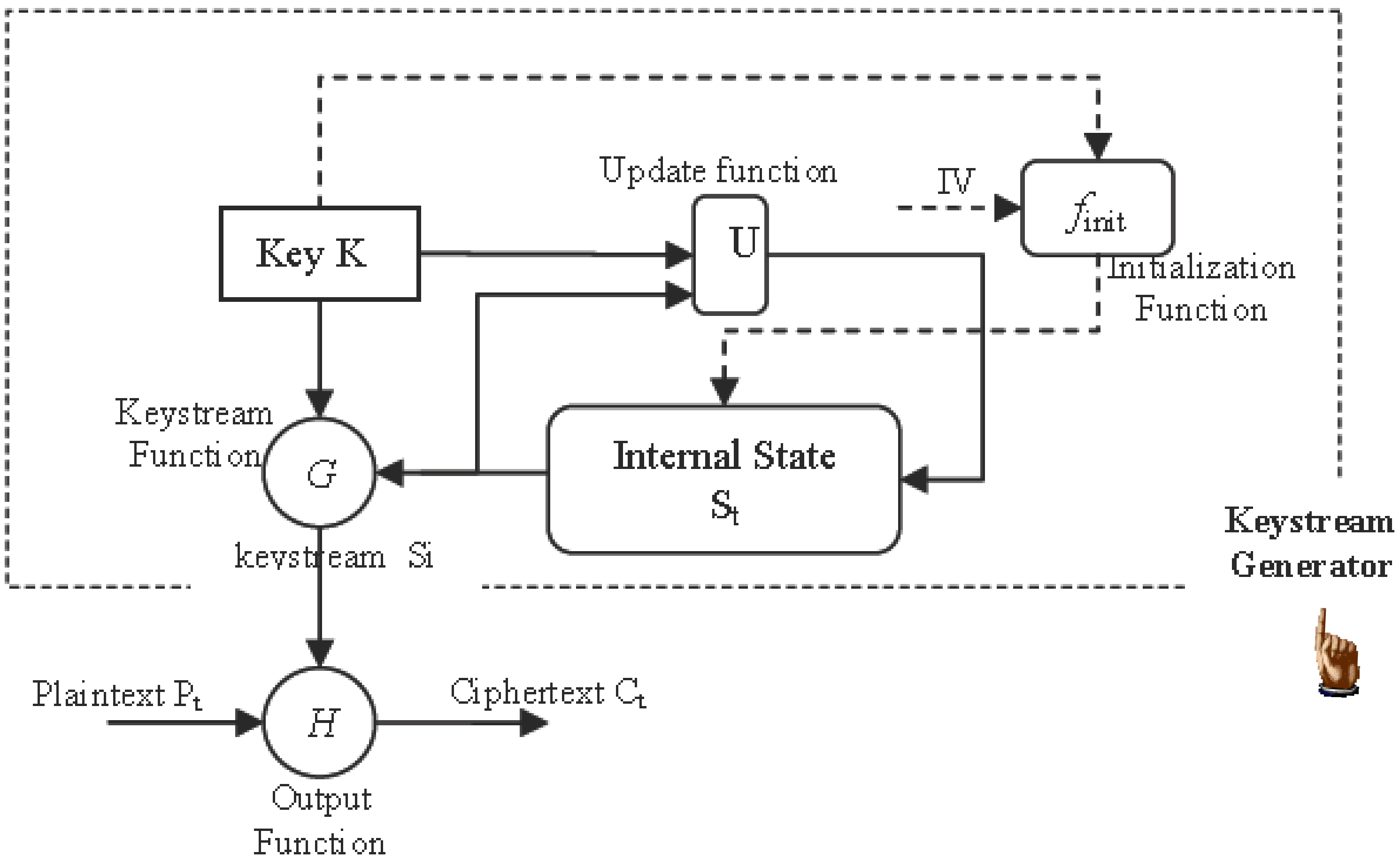
# Key Stream Generation Phase

❖ General Structure of the **synchronous stream cipher**



# Key Stream Generation Phase

❖ General Structure of the **synchronous stream cipher**



# Properties of Synchronous Stream Cipher

# Properties of Synchronous Stream Cipher

❖ Synchronous stream cipher exhibit several properties

1. **No error propagation:** There is **no chance of error propagation** in case of a synchronous stream cipher.
2. **Better security:** **Chosen plaintext/ciphertext attacks** cannot be applied to synchronous stream ciphers. This feature significantly **reduces the security risks.**

# Properties of Synchronous Stream Cipher

- Synchronous stream cipher exhibit several properties

1. **No error propagation:** There is **no chance of error propagation** in case of a synchronous stream cipher.
2. **Better security:** **Chosen plaintext/ciphertext attacks** cannot be applied to synchronous stream ciphers. This feature significantly **reduces the security risks.**

# Desirable Properties of Key Stream Generator

# Properties of Synchronous Stream Cipher

- **Keystream generators** are the **integral part** of the synchronous stream ciphers.
  - ✓ The **security of stream ciphers** depends mostly on the **security features of these generators.**

# Properties of Synchronous Stream Cipher

- **PERIOD**

  - A **keystream generator** is called **periodic**, if after a **specific number of iterations**, it **generates the same sequence again** or come in the same state.

$$S_{t+p} = S_t \text{ Where } t>0,\ p>0$$

  - The **smallest value of p** is called the **period of the generator**.

# Properties of Synchronous Stream Cipher

- Synchronous stream ciphers are **periodic in nature.**
  - ✓ It implies that the **same key will be used** to encrypt **two different messages**
  - ✓ But, it violate the **principle of the One Time Pad (OTP)** and the cipher becomes **susceptible to attack.**
  - ✓ Therefore the **period of the cipher** or the intrinsic keystream generator should be **substantially large for a good cipher.**

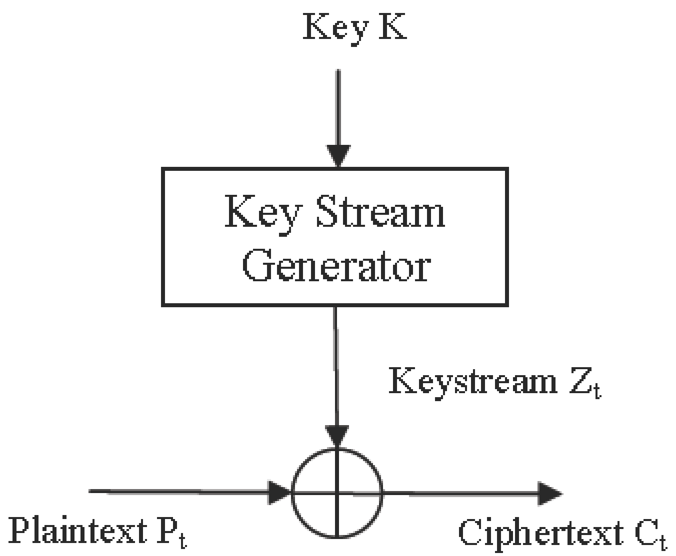
# Properties of Synchronous Stream Cipher

- **RANDOMNESS:**
  - ✓ The **output sequence** should behave like a **truly random stream**
  - ✓ The **output sequence** should be **unpredictable and uniform** i,e 0’s 1's should be equally distributed and with **any given sequence next bits cannot be determined.**

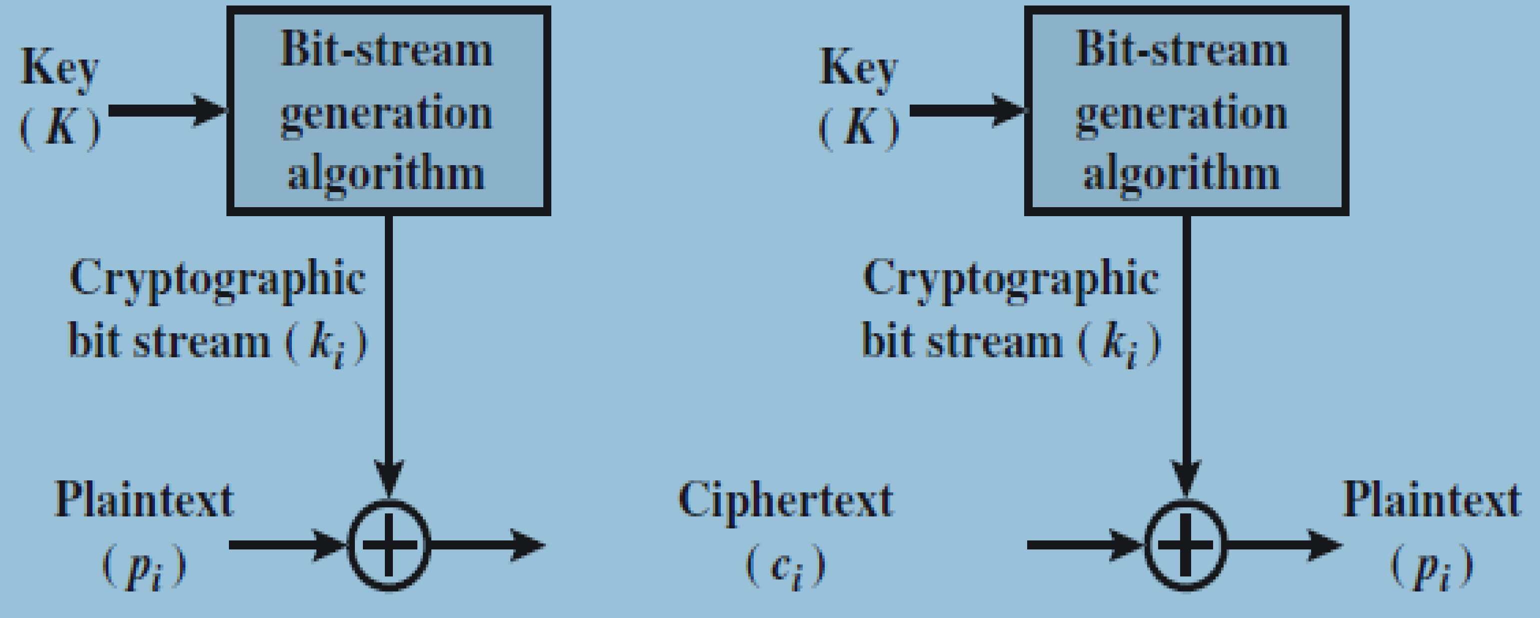
# Binary Additive Stream Cipher

# Binary Additive Stream Cipher

❖ A **binary additive stream cipher** is a **synchronous stream cipher** in which the **keystream, plaintext, and ciphertext digits** are **binary digits**, and the **output function H** is the **XOR (⊕) function.**



# Representation of Binary Additive Stream Cipher




(a) Stream cipher using algorithmic bit-stream generator

# Basic Building Blocks For Synchronous Stream Cipher

# Basic Building Blocks For Synchronous Stream Cipher

❖ Some of the **main building blocks** for synchronous stream cipher design

1. **Feedback Shift Register**
2. **Linear Feedback Shift Register**
3. **Nonlinear Feedback Shift Register**
4. **Feedback Shift Register with Carry**
5. **Boolean Function**
6. **S-Box**

# Feedback Shift Register

# Feedback Shift Register

❖ The **states** in a **feedback shift register (FSR)** can be viewed as values and **stored in a register:**

$$S_t = \{S_0, S_1, \ldots\ldots\ldots\ldots, S_{t-1}\}$$

# Feedback Shift Register

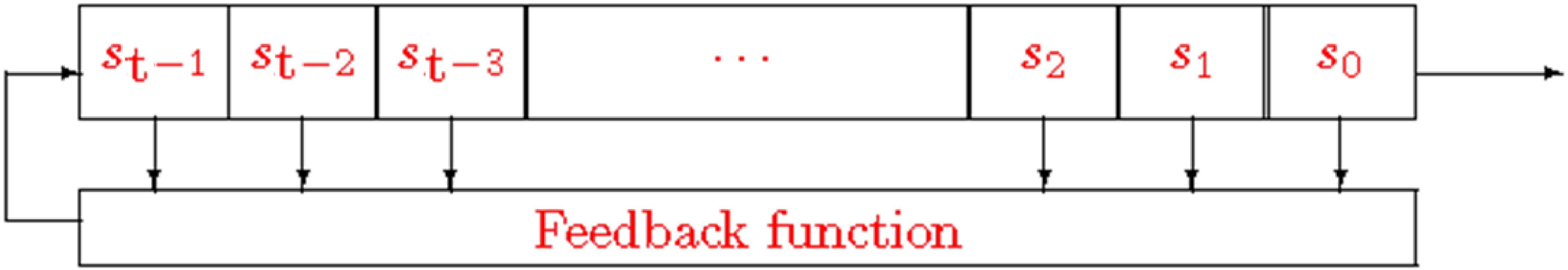
- During **Updation,**
  - ✓ A **new value $S_t$** is calculated using **connection logic or a Boolean function(Feedback Function).**
  - ✓ At every clock, the **values of the register are shifted one bit to the left** and **new value $S_t$ is fed to the rightmost bit.**

# Feedback Shift Register

❖ The **Update Function** can be defined as follows:

$$S_{t+1} = \{S_t, S_0, S_1, \ldots\ldots\ldots\ldots, S_{t-1}\}$$

# Feedback Shift Register



# Types of Feedback Shift Register

- The **feedback shift registers** are classified on the basis of their **transformation or update function.**

1. If the **transformation function** is **LINEAR** then **FSR is termed as Linear Feedback Shift Register (LFSR)**
2. if the **transformation function is NONLINEAR**, the **FSR is termed as Non Linear Feedback Shift Register** (NFSR or NLFSR).

# Linear Feedback Shift Register

# Linear Feedback Shift Register

- **LFSR's** have been widely used in **stream cipher design**
  - ✓ It provides **good statistical properties**
  - ✓ It provides the **long period output** that are suitable for cryptographic purpose.
  - ✓ Its structure can be analyzed using **algebraic techniques**

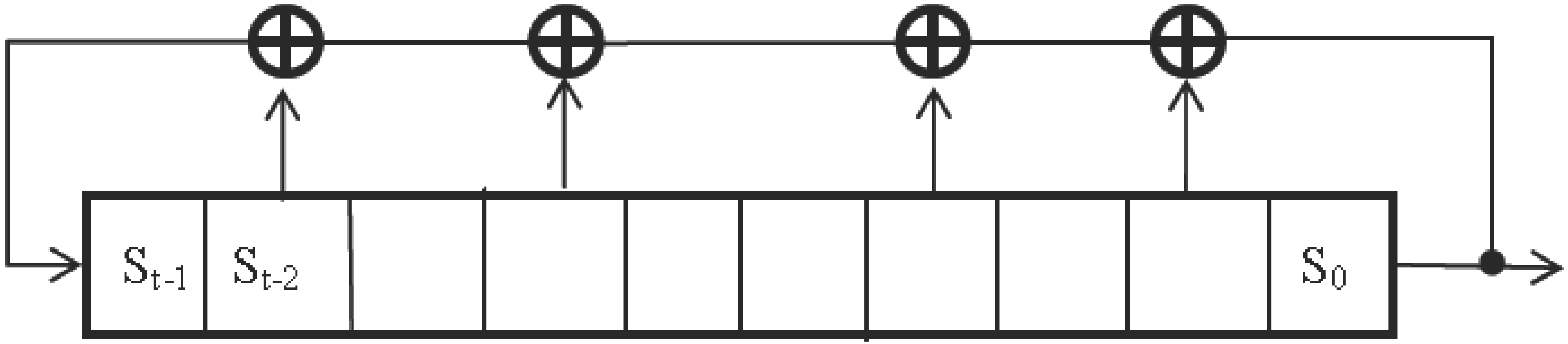
# Linear Feedback Shift Register

- If $F_q$ is defined as a **finite field** which has **q elements** then
  - ✓ An **LFSR of n bits** can be defined as a collection of **n memory cells**

    $m_0, m_1, m_2, \ldots\ldots\ldots, m_{n-1}$
  - ✓ Each have **any value** from $F_q$.

# Linear Feedback Shift Register

❖ The **general structure** of an LFSR



# State of an LFSR

- The **state of an LFSR** is the **content** of that register at any **instance t** and denoted as:

$$S_t\ (S_{t+n-1},\ S_{t+n-2},\ \ldots\ldots\ldots\ldots,\ S_t)$$

# Feedback Polynomial in LFSR

- ❖ LFSR's also use **feedback polynomial of degree n** to **update the LFSR contents:**

$$F(x) = 1+c_1x+ c_2x_2+ \ldots\ldots\ldots\ldots+c_nx_n$$

# Update the State using Feedback

- The **polynomial functions** are **xored (⊕)** with the **contents of the register** and **output bit is generated** that is **again fed in the last bit of the register.**
- When the **LFSR is clocked**, the **$S_0$ is taken as output** and **all the values of the shift register are shifted one bit left** and **the last cell is updated** with the **help of feedback polynomial.**

# New State Value of an LFSR

❖ The **new value of last cell** is calculated as:

$$S_{t+n} = C_n . \sum_{i=0}^{n-1} C_i . S_{t+i} \text{ Over the field } \mathbb{F}_q$$

❖ Where $\mathbf{c_0 + c_1 + \ldots\ldots\ldots\ldots + c_t} \in F_q$ are called **feedback coefficients**.

# Irreducible Polynomial in LFSR

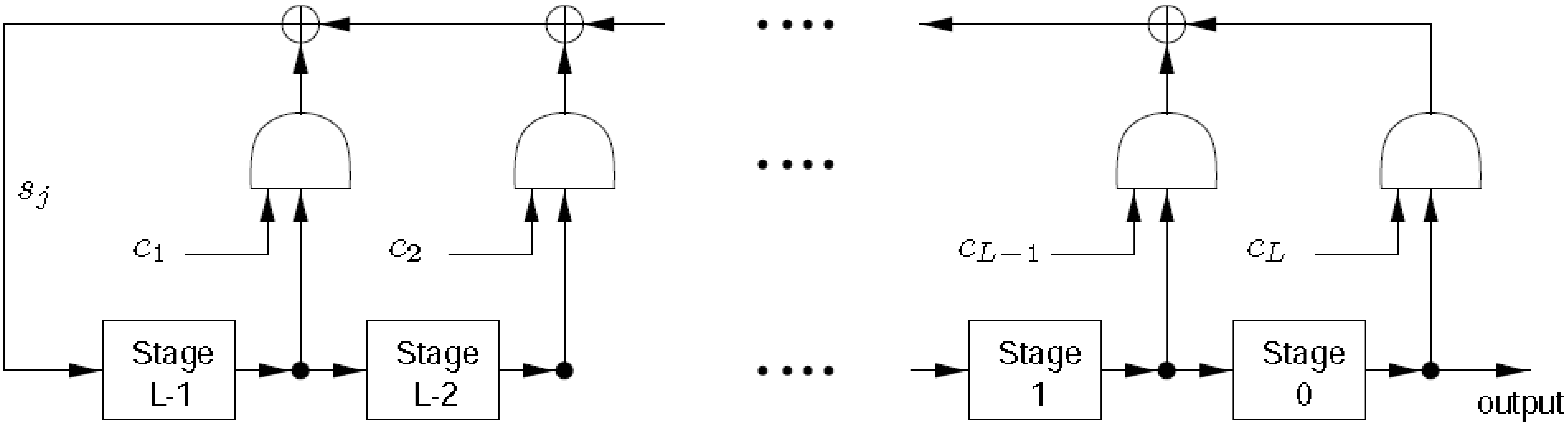
- An **LFSR** which uses **irreducible polynomial** as **feedback polynomial**
  - ✓ **For updation** with **maximum length LFSR period $q^n$-1**
  - ✓ The output sequence of the LFSR can be **m-sequence.**

# Finite Field in LFSR

- We use finite field **$F_2$ or GF(2)** in computers for representation of **binary bits 0 and 1**
  - ✓ **Addition and multiplication** are equivalent to **binary operations XOR & AND.**

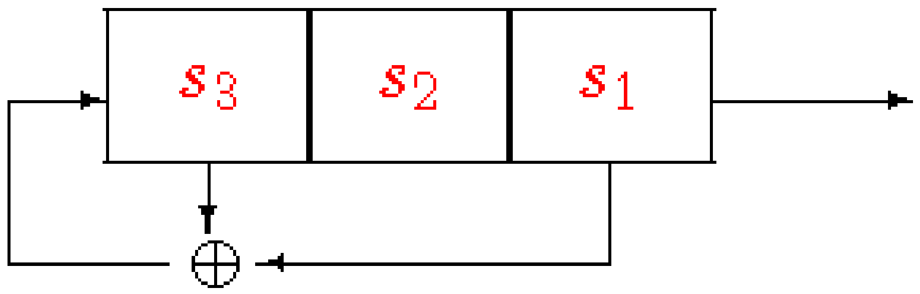
# Addition and multiplication in LFSR

❖ **Addition and multiplication** are equivalent to **binary operations XOR & AND.**



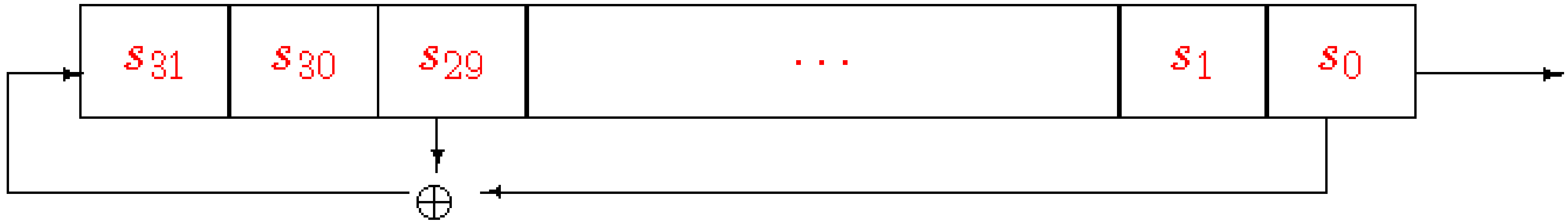
# For Example : LFSR Register: $X^3 + X + 1$

- **LFSR connection polynomial** is given by **$X^3+X+1$**



# For Example : LFSR Register: $X^{32} + X^3 + 1$

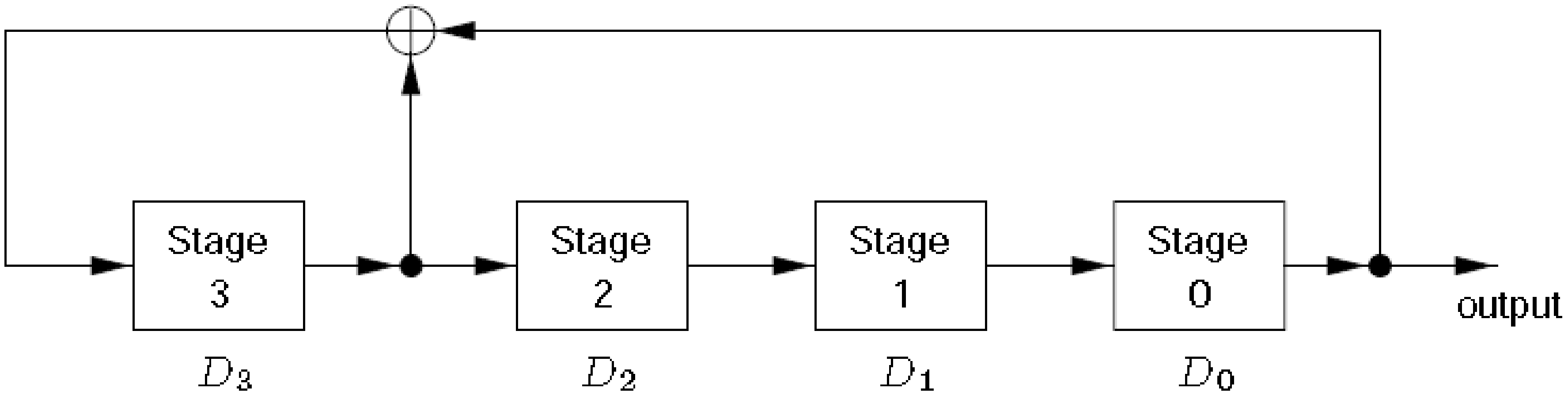
❖ **LFSR connection polynomial** is given by $X^{32} + X^3 + 1$



# For Example : LFSR Register: $X^4 + X + 1$

❖ Consider the **LFSR with polynomial $X^4 + X + 1$**. and If the **initial state of the LFSR is [0; 1; 1; 0],**

❖ Number of contents stages will be **$D_3$, $D_2$, $D_1$, $D_0$**



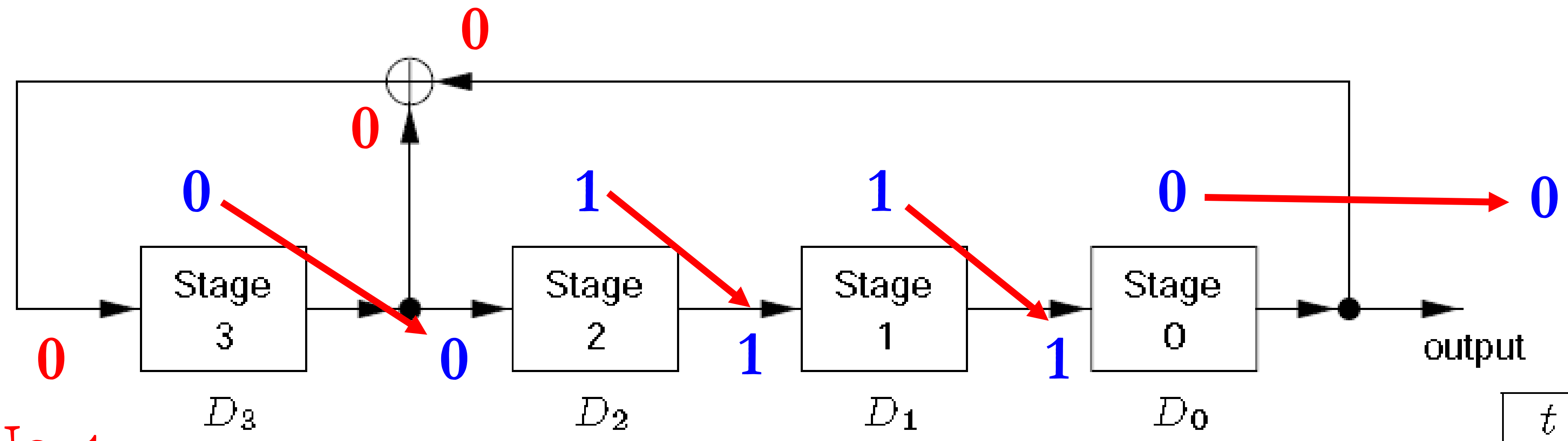
# For Example : LFSR Register: $X^4 + X + 1$

❖ The following tables show the **contents of the stages** **$D_3$, $D_2$, $D_1$, $D_0$** at the end of **each unit of time 't'** , when the initial state is **[0; 1; 1; 0]**,

| $t$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 2 | 1 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 1 |
| 6 | 1 | 0 | 0 | 0 |
| 7 | 1 | 1 | 0 | 0 |

| $t$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|
| 8 | 1 | 1 | 1 | 0 |
| 9 | 1 | 1 | 1 | 1 |
| 10 | 0 | 1 | 1 | 1 |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 0 | 1 | 0 | 1 |
| 13 | 1 | 0 | 1 | 0 |
| 14 | 1 | 1 | 0 | 1 |
| 15 | 0 | 1 | 1 | 0 |

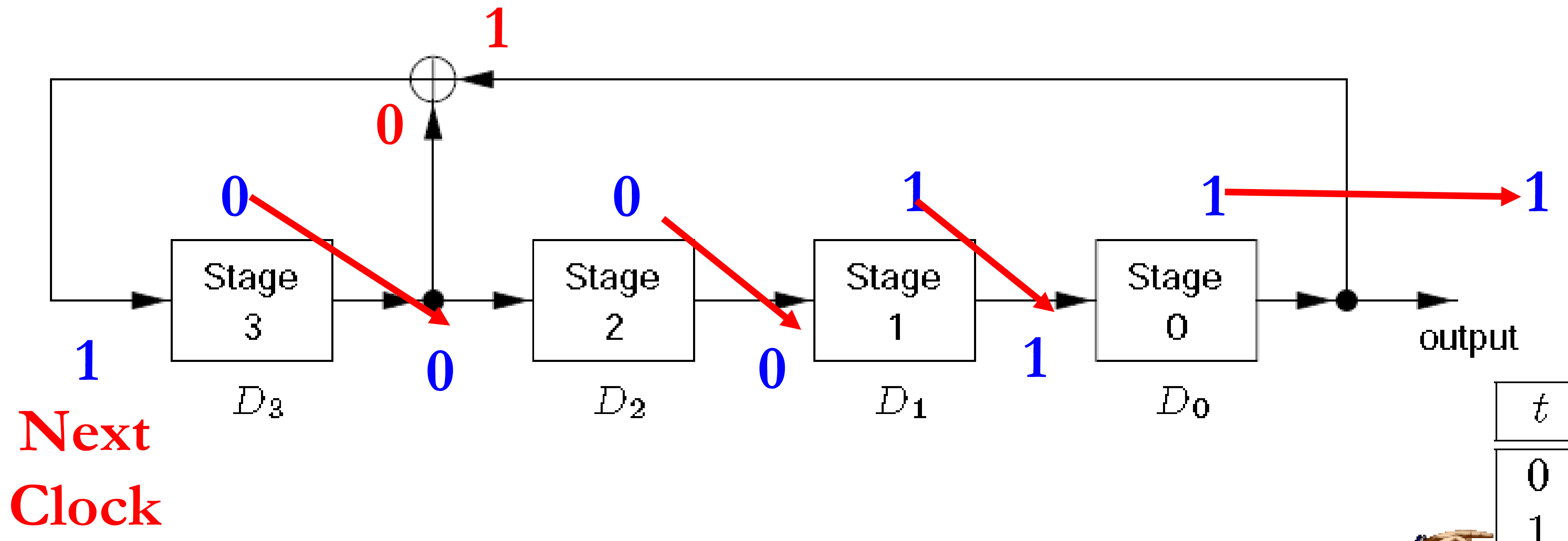
# For Example : Initial Stage




Next Clock

| $t$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 2 | 1 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 1 |
| 6 | 1 | 0 | 0 | 0 |
| 7 | 1 | 1 | 0 | 0 |

# For Example : t = 1




| $t$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 2 | 1 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 1 |
| 6 | 1 | 0 | 0 | 0 |
| 7 | 1 | 1 | 0 | 0 |

**Repeat the Process up to $2^4$**

# Finally We get

| $t$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 2 | 1 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 1 |
| 6 | 1 | 0 | 0 | 0 |
| 7 | 1 | 1 | 0 | 0 |

| $t$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|
| 8 | 1 | 1 | 1 | 0 |
| 9 | 1 | 1 | 1 | 1 |
| 10 | 0 | 1 | 1 | 1 |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 0 | 1 | 0 | 1 |
| 13 | 1 | 0 | 1 | 0 |
| 14 | 1 | 1 | 0 | 1 |
| 15 | 0 | 1 | 1 | 0 |

The output sequence is $s = 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, \ldots$, and is periodic with period 15

# Observation

- **Every output sequence (i.e., for all possible initial states)** of an **LFSR is periodic** if and only if the **irreducible polynomial C(D) has degree n.**

# Advantages in LFSR

- **LFSR's** are used in cryptography due to **large periods** that increases exponentially with the size of LFSR
- Sequences produced by **LFSR show good uniform statistical properties**.
  - ✓ LFSR's based **keystream generators provide large period**

# Disadvantages in LFSR

- LFSR has **low linear complexity**
  - ✓ The **output sequence** is easily predictable using the **Berlekamp - Massey algorithm.**
- Many different techniques have been used to **overcome the weakness of low linear complexity.** Some of these are:
  1. **Non Linear Combination Generator**
  2. **Nonlinear Filter Generator**
  3. **Clock controlled generators**

# Outline

- Symmetric Algorithm Works
- Stream Cipher
- Pseudorandom Number Generation
  - Types of Random Number
- Types of Stream Ciphers
  - Asynchronous or Self Synchronizing Stream Cipher
  - Synchronous Stream Cipher
    - Linear Feedback Shift Register

# Thank U