41900 – Fundamentals of Security

PRNG & Block Cipher

Ashish Nanda

Ashish.Nanda@uts.edu.au

Pseudorandom Number Generators

A source of random numbers is essential in cryptography. We need them for:

- Session Keys
- Shuffling of Cards
- Challenges
- Nonce

Computers are inherently deterministic.

- As a result, true randomness is a difficult thing to come by.

Pseudorandom number generator functions (PRNGs)

- These are used generate what appears to be statistically random output.

Cryptographically secure pseudo random number generator (CSPRNG)

- It is a type of PRNG whose properties make it suitable for use in cryptography.

Sourcing Randomness

PRNG functions produce the same sequence of seemingly random output when provided with particular “**seed**” data.

Since a computer is deterministic, it must extract randomness (**entropy**) from an external, truly random source. This could be something like:

- Thermal noise of hard drives
- Low-order bit fluctuations of voltage readings
- User input
- Geiger counter click timing

Properties of PRNGs

Desirable properties of PRNGs include:

- Repeatability
- Statistical randomness
- Long period / cycle
- Insensitive to seeds

PRNGs can be broken by:

- Statistical tests that find patterns or biases in the output sequence
- Inferring the state of the internal registers from the output sequence

PRNGs are a **critical part of a cryptosystem** as they can often be a single point of failure.

PRNG: Linear Congruential Generators

An **LCG** generates a sequence $\mathbf{x_1, x_2, . . .}$ by starting with a **seed** $\mathbf{x_0}$ and using the rule:

$$\mathbf{x_{n+1} = (ax_n + b) \bmod c}$$

where **a**, **b**, and **c** are fixed constants.

- The **period**(range) of the PRNG is at most **c**.
- Only two values $\mathbf{x_i}$, $\mathbf{x_{i+1}}$ are needed to determine **a** and **b**.
- Commonly found in libraries, e.g. the **Unix rand()** function.
- Must not be used for security purposes – it's easily predictable.

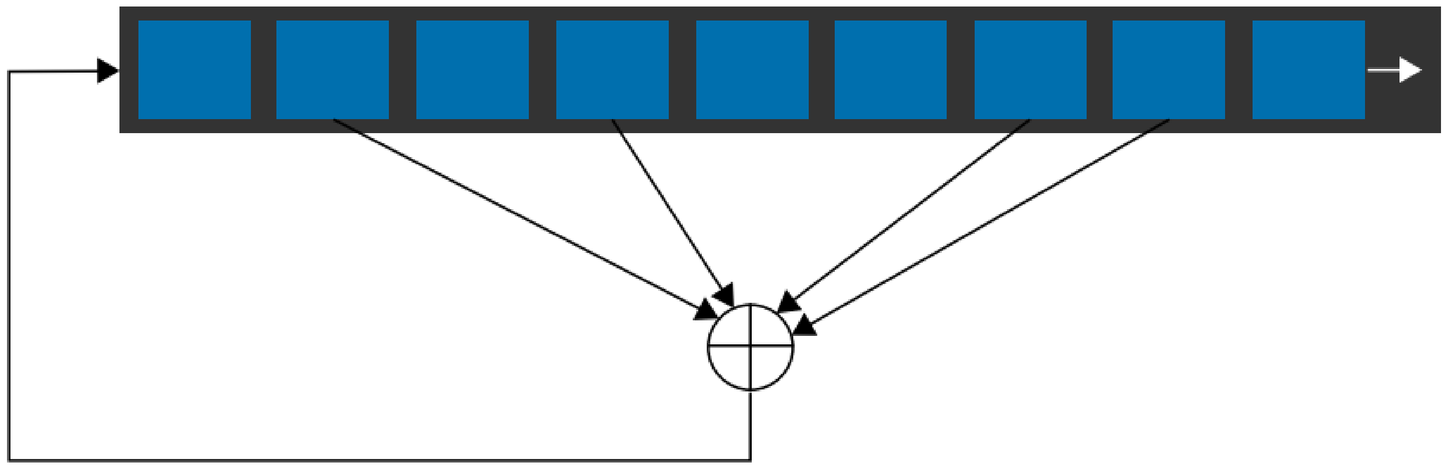
Fun Fact: An LCG was once used by an online casino who were so sure of their code that they published their algorithms….. The results were as one would expect.

PRNG: Linear Feedback Shift Registers

A Linear Feedback Shift Register (**LFSR**) simply combines the bits of a series of registers, and shifts the output onto the register.

- The **seed** is the initial value of the register.
- Easy and fast for hardware (1 bit per clock).

- **Problem:** The configuration can be determined from **2^n** output bits, where **n** is the length of the **LFSR period**.



PRNG: RC4 Stream Cipher

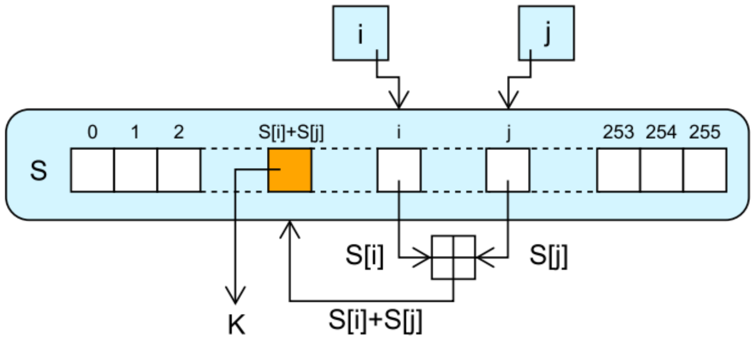
RC4 is a stream cipher which has wide applications in cryptography.

- At one point, RC4 was used to encrypt > 50% of all SSL traffic.
- It is the core algorithm of Wired Equivalent Privacy (**WEP**)

Based on permutations of a 256 byte array, the **seed** is the initial array value.

- RC4's key scheduling algorithm has known problems (e.g. WEP weakness)

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    K := S[(S[i] + S[j]) mod 256]
    output K
endwhile
```



Other PRNGs

ANSI X9.17
Based on 3DES

DSA PRNG
Based on SHA or DES

RSAREF PRNG
Based on MD5 hashing and addition modulo 2128

Using PRNGs

- Be extremely careful with PRNG seeds!
- Hash PRNG inputs with a timestamp or counter
- Reseed the PRNG occasionally
- Use a hash function to protect PRNG outputs if PRNG is suspect

XOR and OTP

What is XOR?

XOR is the “exclusive or” operation.

That means: one or the other, but not both.

It is addition modulo 2 and is represented by $\oplus$.

$$a \oplus b = (a + b) \bmod 2$$

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

XORing Bits

Typically we XOR bits together

- Often the plaintext will be XOR'd with a key stream to produce ciphertext.
- This is effectively the same as a Vigenère cipher.
- Where a XOR is addition modulo 2, a Vigenère cipher is addition modulo 26 since XOR works with bits and not letters.

XOR Encryption

```
Plaintext:   0110110001101111101101100
Keystream:   0110001101100001011110100
=========================================
Ciphertext:  0000111100001110000011000
```

Interesting XOR Properties

Anything **XOR'd with itself is zero**.

$$\mathbf{A \oplus A \equiv 0}$$

XOR is **Associative**:

$$\mathbf{A \oplus (B \oplus C) \equiv (A \oplus B) \oplus C}$$

XOR is **Commutative**:

$$\mathbf{A \oplus B \equiv B \oplus A}$$

XOR

XOR (addition modulo 2) is commonly used to provide security in programs.

It is very weak by itself, but forms the building block of most crypto primitives.

The message **m** is XOR'd bitwise with a secret key:

$$\mathbf{c = m \oplus k}$$

$$\mathbf{m = c \oplus k}$$

XOR

XOR is effectively a Vigenère cypher and easy to break:

- Determine the key length **N** from index of coincidence
- Shift cyphertext by **N** and XOR with itself

This removes the key ($\mathbf{c \oplus c' = m \oplus k \oplus m' \oplus k = m \oplus m'}$)

- Results in message XOR’d with a shifted version of itself
- Language is extremely redundant
- Easy to then decrypt

One Time Pad (OTP)

A **one time pad** is using a different substitution cipher for each letter of the plaintext.

A one time pad is **perfectly secure** provided that:

- The secret key **k** is truly random
- The plaintext does not repeat
- The keystream does not repeat

Failure to meet any one of these requirements results in **zero security**.

One Time Pad (OTP)

The strength comes from the fact that a truly random key added to plaintext, produces a truly random ciphertext.

No amount of computing power can break a one time pad. Brute force would yield each and every possible message that length.

Core Problems: key distribution, key destruction, synchronisation.

- **k** must be same length as **m**:
  to encrypt 1GB you need a 1GB shared key.
- Used for ultra-secure, low bandwidth communications
  e.g. military satellites, Moscow-Washington phone line
- Future: Quantum Key Distribution
  secure distribution at a distance.

Perfect Secrecy

Goal of cryptography: **Ciphertext reveals nothing about the plaintext**.

A cipher has perfect secrecy if, for all **m ∈ M**, **c ∈ C**, the plaintext and ciphertext are statistically independent:

$$\mathbf{Pr[m_1 = m_2 | c_1 = c_2] = Pr[m_1 = m_2]}$$

Assuming each transmitted message is equally likely, the probability that the transmitted message is **m** is:

$$\mathbf{Pr[m1 = m2] = |M|^{-1}}$$

Now the probability that the transmitted message is m given that the observed ciphertext is **c** is:

$$\mathbf{Pr[m_1 = m_2] = \frac{|\{k: E_k(m) = c, k \in K\}|}{|k|}}$$

Perfect Secrecy

The key space **K** must be at least as large as the set of plaintexts:

$$|k| \geqslant |M|$$

For **M = C = {0, 1}^{n}** any cipher with perfect secrecy satisfies $|K| \geqslant 2^n$

The one time pad has perfect secrecy as: **M = C = {0, 1}^{n}** thus:

$$\mathbf{Pr[m_1 = m_2] = \frac{1}{2^n}}$$

$$\mathbf{Pr[m_1 = m_2 | c_1 = c_2] = \frac{1}{2^n}}$$

Note: we require **k ∈ K** to be as long as the message, which means we need to securely communicate a key as long as the message in advance.

Breaking OTP: Two Time Pad

A **two-time pad** is **perfectly insecure**. Suppose two messages $\mathbf{m_1}$, $\mathbf{m_2}$ are encoded using the same key $\mathbf{k}$:

$\mathbf{c_1 = m_1 \oplus k}$

$\mathbf{c_2 = m_2 \oplus k}$

Then the key $\mathbf{k}$ may be cancelled by XORing the ciphertexts:

$$\begin{aligned} \mathbf{c_1 \oplus c_2} &= \mathbf{(m_1 \oplus k) \oplus (m_2 \oplus k)} \\ &= \mathbf{m_1 \oplus m_2 \oplus k \oplus k} \\ &= \mathbf{m_1 \oplus m_2} \end{aligned}$$

$\mathbf{m_1 \oplus m_2}$ is easy to separate due to the redundancy in English and in ASCII (for example, bit 6 is set in letters but not most punctuation).

Breaking OTP: Malleability

The OTP and all stream ciphers are **highly malleable**. Suppose plaintext is a one bit vote **v ∈ 0, 1**

- **v = 0** is a vote for Labor
- **v = 1** is a vote for Liberal

Alice encrypts her vote using OTP and sends to Bob:

$$\mathbf{c = v \oplus k} \text{ where } \mathbf{k \in 0, 1} \text{ is randomly chosen}$$

Mallory intercepts the ciphertext and sends with bits flipped:

$$\mathbf{c' = c \oplus 1}$$

Bob receives c’ and decrypts vote:

$$\begin{aligned} \mathbf{c' \oplus k} &\mathbf{= c \oplus 1 \oplus k} \\ &\mathbf{= v \oplus k \oplus 1 \oplus k} \end{aligned}$$

Block Cipher Modes of Operation

Cipher Modes of Operation

Once a key **k** is chosen and loaded into a block cipher, $\mathbf{E_k}$ only operates on single blocks of data.

1. Block size usually small (**16 byte** blocks for AES)
2. Message to be sent usually large (web page + assets ≈ 500kB)
3. Need a way to repeatedly apply the cipher with the same key to a large message.

By using different modes of operation, messages of an arbitrary length can be split into blocks and encrypted using a block cipher.

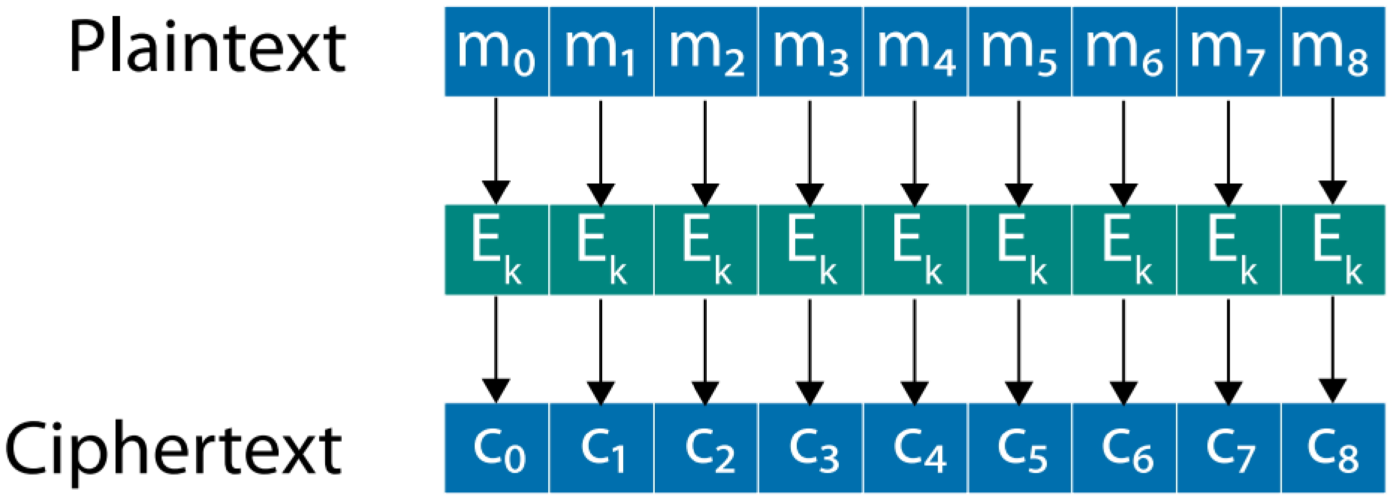
Each mode of operation describes how a block cipher is repeatedly applied to encrypt a message and each has certain advantages and disadvantages.

Evaluating Block Ciphers & Modes

To evaluate a cipher and a mode of operation, examine:

- **Key Size**: Upper bound on security, but longer keys add costs (generation, storage, etc.)
- **Block Size**: Larger is better to reduce overheads, but is more costly.
- **Estimated Security Level**: Confidence grows the more it is analysed.
- **Throughput**: How fast can it be encrypted/decrypted? Can it be pre-computed? Can it be parallelised?
- **Error Propagation**: What happens as a result of bit errors or bit loss?

The first two points above are relevant only to the cipher, while the last three are relevant to both the cipher and a mode of operation.

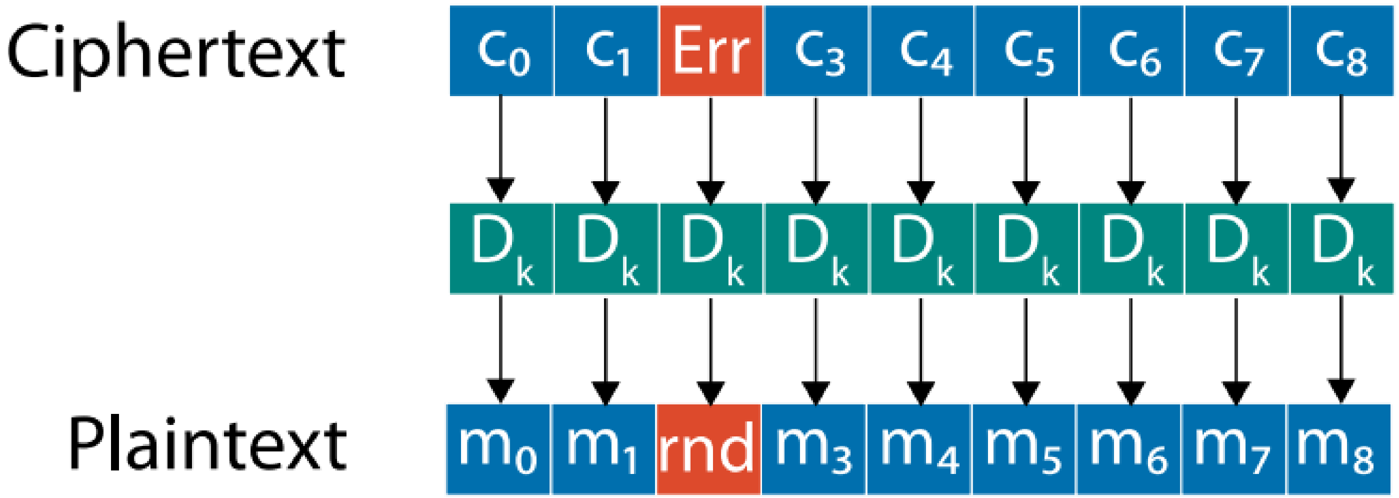


Electronic Code Book (ECB) encrypts each block separately.

ECB is generally an insecure and naïve implementation.

It is vulnerable to a range of attacks including dictionary and frequency attacks.

Electronic Code Book (ECB)

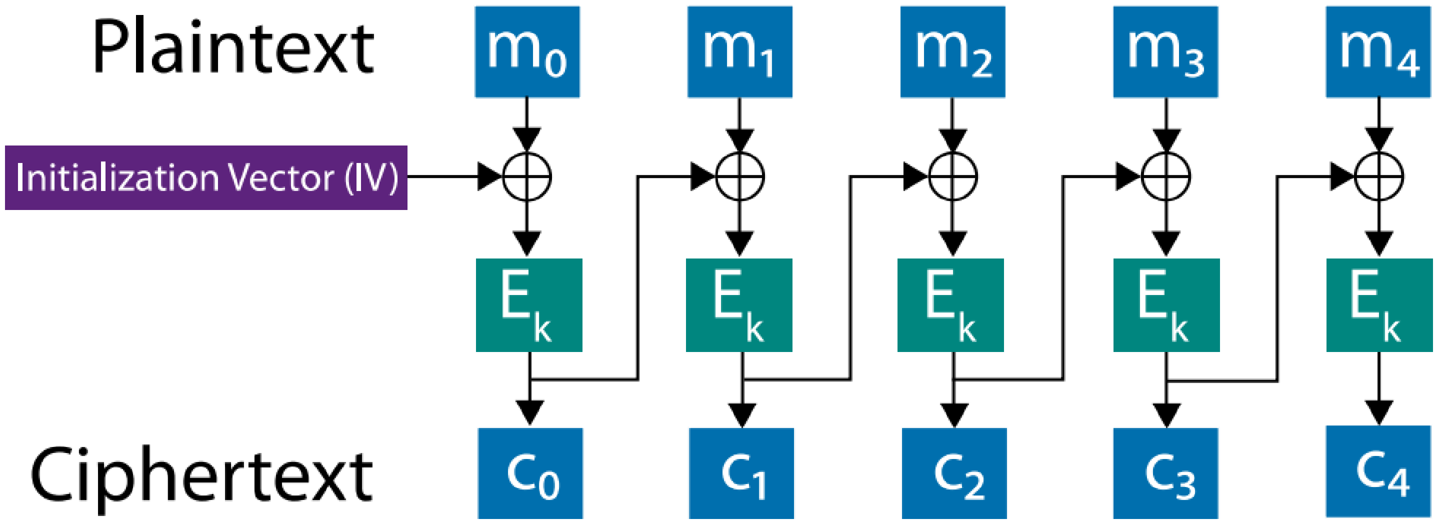


Identical plaintext blocks result in identical ciphertext blocks

- Since blocks are enciphered independently, a reordering of ciphertext blocks results in reordering of plaintext blocks.
- ECB is thus not recommended for messages > 1 block in length.

Error propagation: Bit errors only impact the decoding of the corrupted block.

ECB Properties



In **Cipher Block Chaining (CBC)** blocks are chained together using XOR.

The **Initialization Vector (IV)** is a random value that is transmitted in the clear that ensures the same plaintext and key does not produce the same ciphertext.

Cipher Block Chaining (CBC)

CBC Properties

Identical plaintexts result in identical ciphertexts when the same plaintext is enciphered using the same **key** and **IV**. Changing at least one of **[k, IV, m_0]** affects this.

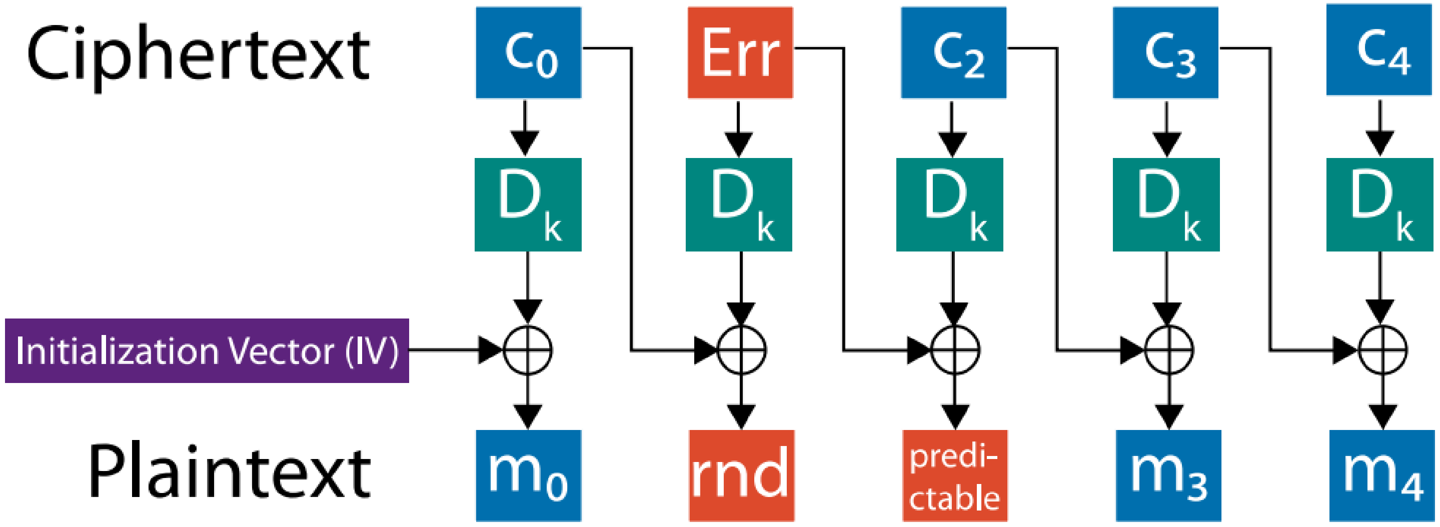
- Rearrangement of ciphertext blocks affects decryption. As ciphertext part $\mathbf{c_j}$ depends on all of **[m_0, m_1, · · · , m_j]**.

Error propagation:

- Bit error in ciphertext $\mathbf{c_j}$ affects deciphering of $\mathbf{c_j}$ and $\mathbf{c_{j+1}}$. Recovered block $\mathbf{m_j}$ typically results in random bits.
- Bit errors in recovered block $\mathbf{m_{j+1}}$ are precisely where $\mathbf{c_j}$ was in error. Attacker can cause predictable bit changes in $\mathbf{m_{j+1}}$ by altering $\mathbf{c_j}$.

Bit recovery:

- CBC is self-synchronising in that if a bit error occurs in $\mathbf{c_j}$ but not $\mathbf{c_{j+1}}$, then $\mathbf{c_{j+2}}$ correctly decrypts to $\mathbf{m_{j+2}}$.

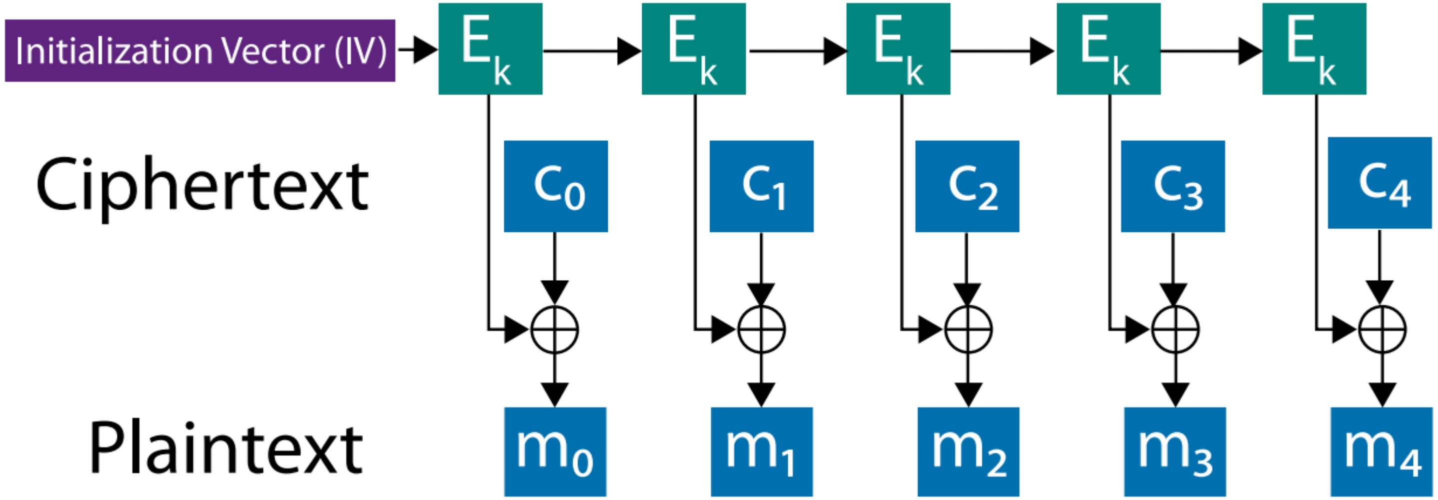


Ciphertext errors only affect two plaintext blocks, one in a predictable way.

Encryption must be done sequentially.

Decryption can be random-access and is fully parallelisable.

CBC Decryption



Output Feedback Mode (OFB)

Effectively turns a block cipher into a synchronous stream cipher.

Output Feedback Mode (OFB)

OFB Properties

Identical plaintext results in identical ciphertext when the same plaintext is enciphered using the same key and IV.

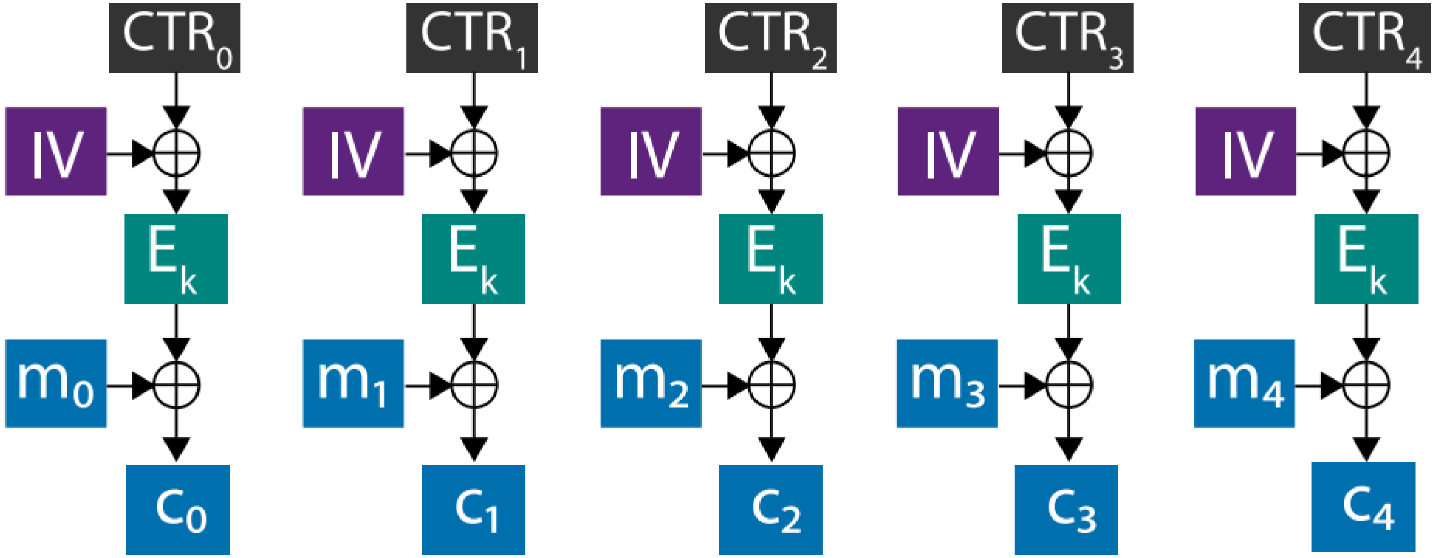
Chaining Dependencies: (Same as a stream cipher) The key stream is plaintext independent.

Error propagation: (Same as a stream cipher) Bit errors in ciphertext blocks cause errors in the same position in the plaintext.

Error recovery: (Same as a stream cipher) Recovers from bit errors, but not bit loss (misalignment of key stream)

Throughput: Key stream may be calculated independently — e.g. precomputed — before encryption/decryption become parallelisable.

IV must change: Otherwise it becomes a two time pad.



Counter Mode (CTR) modifies the IV for each block using a predictable counter function, turning the block cipher into a stream cipher.

The counter can be any function (e.g. a PRNG), but it is commonly just an incrementing integer.

Counter Mode (CTR)

CTR Properties

Identical plaintext results in identical ciphertext when the same plaintext is enciphered using the same key and IV.

Chaining Dependencies: (Same as a stream cipher) The key stream is plaintext independent.

Error propagation: (Same as a stream cipher) Bit errors in ciphertext blocks cause errors in the same position in the plaintext.

Error recovery: (Same as a stream cipher) Recovers from bit errors, but not bit loss(misalignment of key stream)

Throughput: Both encryption and decryption can be randomly accessed and/or parallelised: the best we could hope for.

IV must change: Otherwise it becomes a two time pad.

GCM Mode

Galois/Counter Mode (GCM) mode is not strictly a cipher mode of operation since it also provides authentication: assurance the ciphertext has not been tampered with.

- An extension of CTR mode.
- While encryption happens, the ciphertext blocks are combined into something like a MAC.
- Unlike HMAC, is parallelisable (you can't combine two HMACs into one larger one).
- Used for low-latency, high-throughput dedicated hardware applications (network packets).

GCM mode is an example of authenticated encryption.