Stream and Block Ciphers

DSC 4110/6111: Modern Cryptography and AI/ML Security

Dr. Laltu Sardar

School of Data Science,
Indian Institute of Science Education and Research Thiruvananthapuram (IISER TVM)

August 3, 2026

PRG and Stream Cipher

The Need for a Pseudorandom Generator (PRG)

Recall from the Previous Lecture

Diffie-Hellman Key Exchange (or a PQ Key Encapsulation Mechanism) establishes only a **short shared secret**.
Suppose Alice and Bob wish to encrypt a $10\,\mathrm{GB}$ file using the One-Time Pad.

$$\boxed{\text{10 GB Message} \Longrightarrow \text{10 GB Secret Key}}$$

Generating billions of key bits by repeatedly executing Diffie-Hellman is computationally expensive.

Can we generate a very long secret from one short secret?

Pseudorandom Generator (PRG)

A pseudorandom generator stretches a short random seed into a much longer sequence that appears random to every efficient adversary.

$$G : \{0,1\}^n \to \{0,1\}^\ell, \qquad \ell > n.$$

What Does "Pseudorandom" Mean?

True Random Bits

- Generated from physical sources.
- Expensive to obtain.
- Limited availability.

Pseudorandom Bits

- Generated from a short seed.
- Produced deterministically.
- Computationally indistinguishable from random.

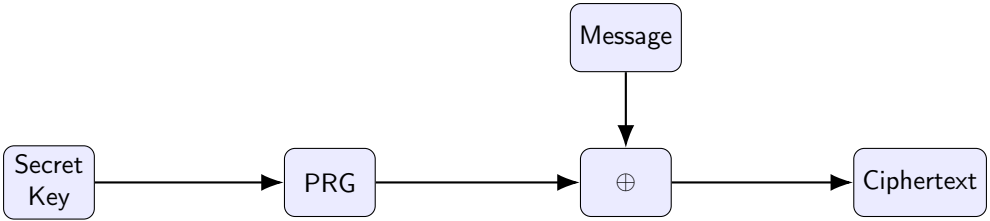
Security Intuition

To every efficient adversary,

$$g(s)$$

should look just like a truly random bit string.

From PRG to Stream Cipher



The PRG output acts as a **keystream**.
Encryption is simply

$$c = m \oplus g(K).$$

A **stream cipher** is therefore a PRG together with the XOR operation.

Therefore, if we have a stream cipher, we can conceptually eliminate the XOR operation and use its keystream generator as a PRG.

Encryption and Decryption

Encryption:

$$c = m \oplus g(k).$$

Decryption:

$$m = c \oplus g(k).$$

since

$$(m \oplus g(k)) \oplus g(k) = m.$$

Key Observation

The same keystream is generated independently by both the sender and the receiver using the shared secret key.

Advantages of Stream Ciphers

Advantages

- Extremely fast.
- Small memory footprint.
- Suitable for long messages.
- Ideal for real-time communication.

Examples

- ChaCha20
- Salsa20
- SNOW 3G
- ZUC

Modern protocols such as TLS, SSH, and VPNs widely employ stream ciphers.

Constructing a PRG from a Hard Problem

Suppose the sender and receiver share a random seed

$$s \in \{0,1\}^n.$$

A computationally hard problem (e.g., the discrete logarithm problem) can be used to derive one pseudorandom bit at a time.

Repeating this procedure produces a long pseudorandom sequence.

$$s \longrightarrow b_1\, b_2\, b_3 \cdots b_\ell$$

Observation

Although theoretically elegant, these constructions are computationally expensive because each output bit (or a few bits) requires solving a complex mathematical operation.

Block Cipher

From Stream Ciphers to Block Ciphers

So far, we have studied stream ciphers, where a pseudorandom generator produces a long keystream.

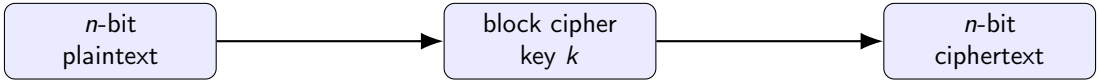
$$c = m \oplus g(k).$$

Question

Can we design a completely different encryption primitive?
Instead of generating a long keystream, can we encrypt one fixed-size block directly?

This motivates the notion of a **block cipher**.

What is a Block Cipher?



A block cipher is a keyed function

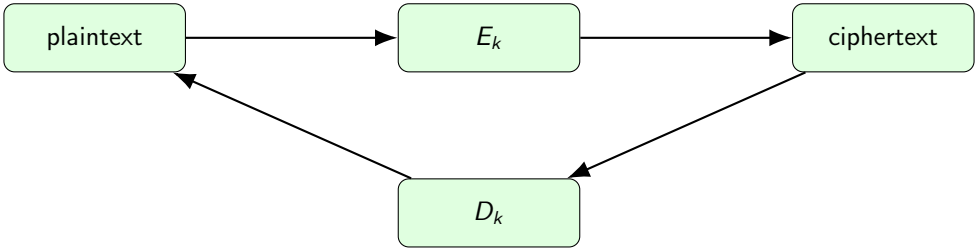
$$E_k : \{0,1\}^n \to \{0,1\}^n.$$

For every key k, the mapping E_k is a permutation over all n-bit blocks.
At present, we consider $n = 128$ to be the minimum.

A Block Cipher is a Keyed Permutation

For every secret key k,

$$D_k(E_k(m)) = m.$$



Unlike a hash function, a block cipher is invertible.
For every key, encryption and decryption are inverse permutations.

AES: The Standard Block Cipher

Advanced Encryption Standard (AES)

Today, AES is the most widely deployed block cipher.

- standardized by NIST
- block size: 128 bits
- key sizes: 128, 192, 256 bits
- efficient in both hardware and software
- deployed in TLS, SSH, VPNs, WiFi, cloud storage, and disk encryption

Block Ciphers

Block Cipher	Block Size	Key Size(s)	Rounds	Key Property
AES	128	128, 192, 256	10, 12, 14	SPN, NIST standard
DES	64	56	16	Feistel, obsolete
3DES	64	112, 168	48	Triple DES, legacy
Blowfish	64	32–448	16	Variable key length
Twofish	128	128, 192, 256	16	AES finalist
Camellia	128	128, 192, 256	18, 24	ISO/IEC standard
Serpent	128	128, 192, 256	32	High security margin
IDEA	64	128	8.5	Mixed algebraic operations
PRESENT	64	80, 128	31	Lightweight SPN
SIMON	32–128	64–256	32–72	Lightweight Feistel
SPECK	32–128	64–256	22–34	ARX-based lightweight

A New Problem

Suppose we wish to encrypt

$$10 \text{ GB}$$

of data.

AES encrypts only

$$128\text{-bit blocks.}$$

Question

How can we securely encrypt a message consisting of millions of 128-bit blocks using the same block cipher?

The answer is to use a suitable

mode of operation.

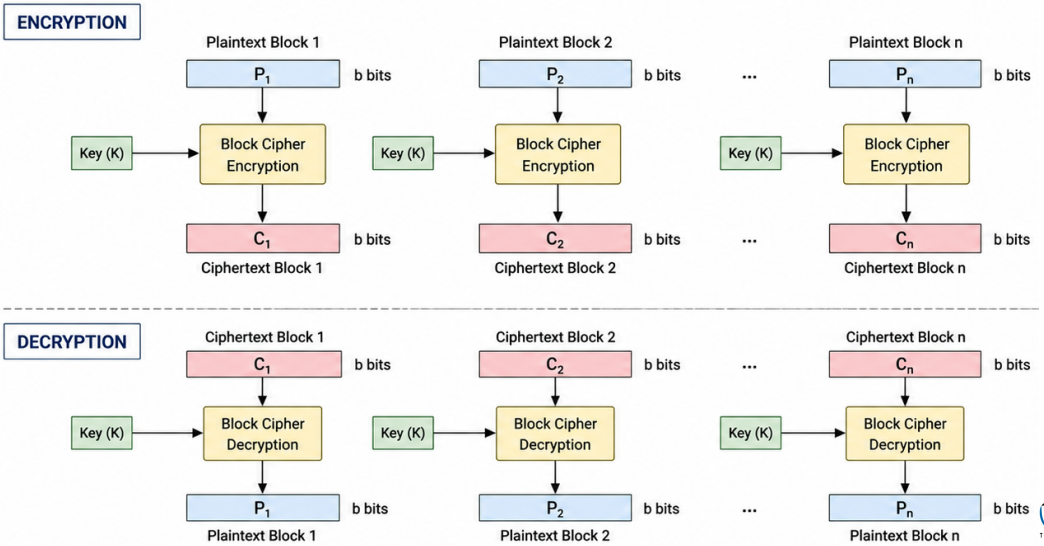
Modes of Operation

The same block cipher can be used with different modes of operation. Some commonly used modes are listed below.

- ECB (Electronic Codebook)
- CBC (Cipher Block Chaining)
- CTR (Counter Mode)

Many other modes of operation also exist, such as CFB (Cipher Feedback), OFB (Output Feedback), GCM (Galois/Counter Mode), XTS, and CCM. Each mode specifies how the block cipher is applied to encrypt data longer than a single block and offers different security, functionality, and performance characteristics.

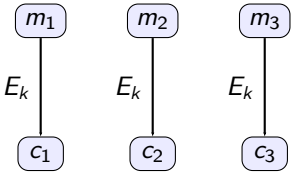
ECB: The Simplest Mode



Why is ECB Problematic?

In Electronic Codebook (ECB) mode, each plaintext block is encrypted independently.

$$c_i = E_k(m_i).$$



If two plaintext blocks are identical, i.e., $m_i = m_j$, then $c_i = c_j$.
Hence, ECB leaks patterns and repeated plaintext blocks.

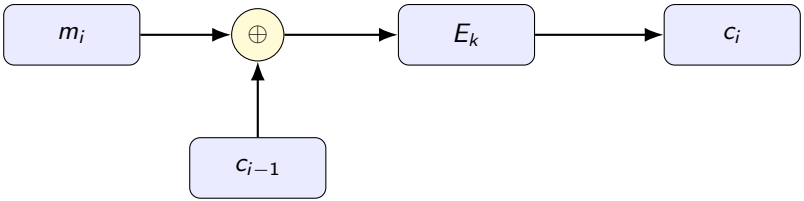
Observation

The input to the block cipher is deterministic. The same plaintext block always produces the same ciphertext block.

How Does CBC Solve This Problem?

CBC randomizes the input to the block cipher before encryption.

$$c_i = E_k(m_i \oplus c_{i-1}), \qquad c_0 = IV.$$

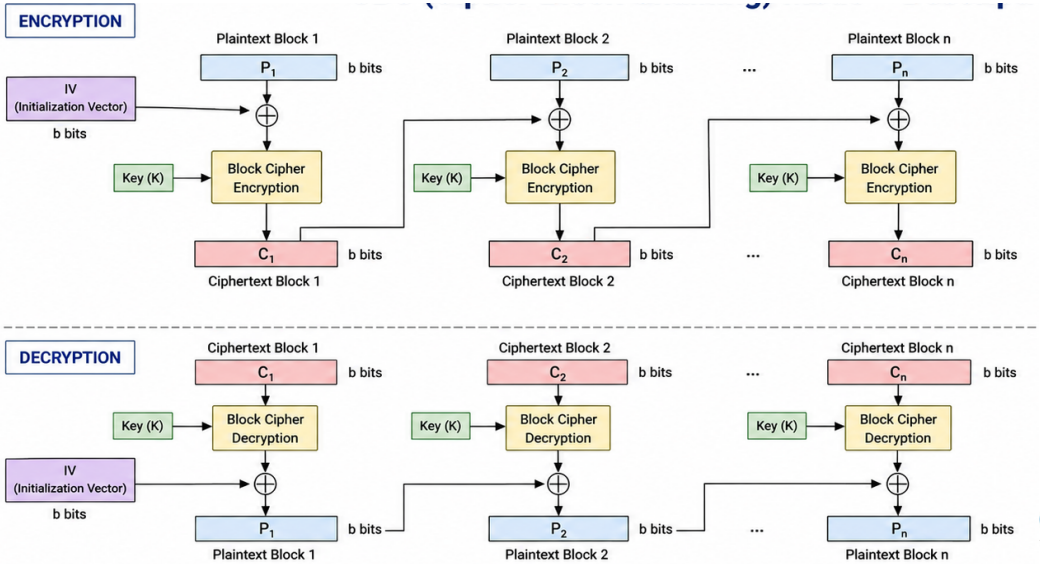


Even if $m_i = m_j$, the inputs to the block cipher become different since $m_i \oplus c_{i-1} \neq m_j \oplus c_{j-1}$ with overwhelming probability.

Key Idea

A fresh random IV makes the first input unpredictable, while each ciphertext block randomizes the input of the next block. Consequently, the same plaintext encrypts to different ciphertexts in different executions.

Cipher Block Chaining (CBC) mode: Encryption & Decryption



Cipher Block Chaining (CBC) Mode

CBC (Cipher Block Chaining)

Before encryption, each plaintext block is XORed with the previous ciphertext block.

$$c_i = E_k(m_i \oplus c_{i-1}), \qquad c_0 = \mathrm{IV}.$$

Thus, every ciphertext block depends on all preceding plaintext blocks, hiding repeated plaintext patterns.

Decryption

$$m_i = D_k(c_i) \oplus c_{i-1}.$$

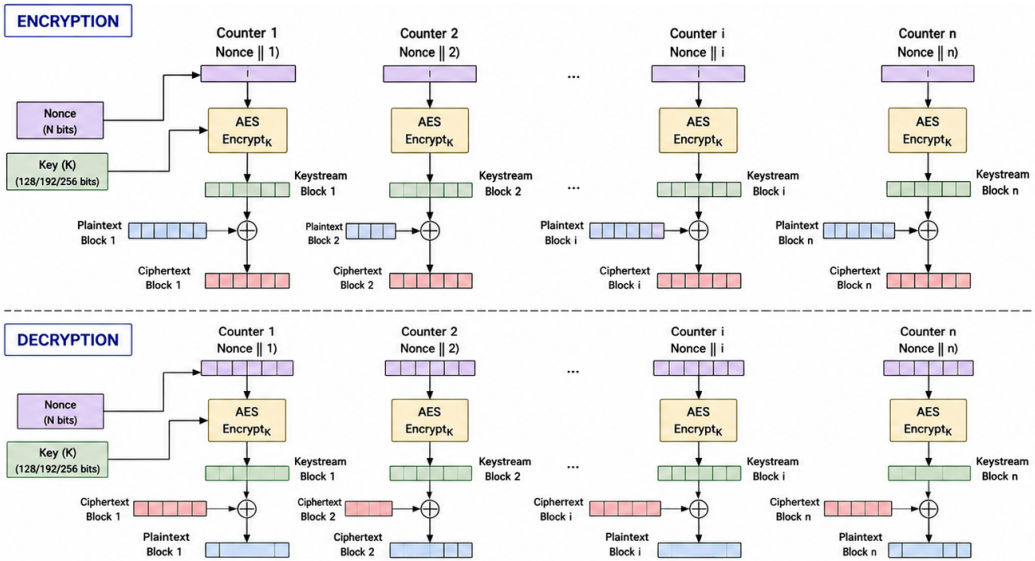
Limitation

Encryption is inherently sequential since computing c_i requires c_{i-1}.

Initialization Vector (IV)

- Fresh and uniformly random.
- Public; need not be secret.
- Never reuse with the same key.
- Different IVs produce different ciphertexts for the same plaintext.

Counter(CTR) Mode: Encryption & Decryption



Counter Mode

CTR (Counter Mode)

The block cipher encrypts a sequence of counters to generate a keystream, which is then XORed with the plaintext.

$$s_i = E_k(\mathrm{ctr} + i), \qquad c_i = m_i \oplus s_i.$$

Since each counter is independent, encryption and decryption are fully parallelizable, and CTR behaves like a stream cipher.

Counter (CTR) Mode

Encryption

$$c_i = m_i \oplus E_k(\text{Nonce} \,\|\, \text{Ctr}_i).$$

Decryption

$$m_i = c_i \oplus E_k(\text{Nonce} \,\|\, \text{Ctr}_i).$$

CTR converts a block cipher into a stream cipher by encrypting successive counter values to generate a pseudorandom keystream.

Nonce

- Must be unique for every encryption.
- Public; need not be secret.
- Never reuse with the same key.
- Reusing a nonce exposes relationships between plaintexts.

Advantages

Encryption and decryption are fully parallelizable, support random access, and the keystream can be precomputed.

Initialization Vector (IV) and Nonce

CBC: Initialization Vector (IV)

- Size = one block (128 bits for AES).
- Chosen uniformly at random for every encryption.
- Sent along with the ciphertext.
- Public, but must never be predictable before encryption.

CTR: Nonce and Counter

- Input to AES: Nonce $\|$ Ctr.
- Total size = one block (128 bits for AES).
- Typical split: 96-bit nonce + 32-bit counter.
- Nonce must be unique for each encryption; counter starts from 0 and increments by 1.

Practical Generation

The sender generates the IV/nonce and transmits it with the ciphertext. The receiver simply reads the transmitted value and reconstructs the same inputs for decryption. Uniqueness is typically ensured using a cryptographically secure random number generator or a monotonically increasing message/packet sequence number.

CBC and CTR Modes

CBC

- each block depends on the previous ciphertext
- hides repeated plaintext blocks
- sequential encryption
- Used mainly in file encryption

CTR

- encrypts counters
- generates a keystream
- fully parallelizable
- random access
- Widely used

CTR is one of the most widely used modes in modern cryptography.

CTR Mode and Stream Ciphers

CTR mode computes

$$s_i = E_k(\text{counter}_i),$$

and encrypts

$$c_i = m_i \oplus s_i.$$

Compare this with a stream cipher

$$c = m \oplus g(k).$$

Key Insight

CTR mode converts a block cipher into a stream cipher.
The encrypted counters play the role of the pseudorandom keystream.

References

Textbook:

1. Douglas R. Stinson and Maura Paterson. *Cryptography: Theory and Practice*. 4th Edition. Chapman & Hall/CRC Press, 2019. ISBN: 9780367148591.

Acknowledgment:

- The figures in this work were generated with the assistance of ChatGPT (OpenAI).

Dr. Laltu Sardar, Assistant Professor, IISER Thiruvananthapuram
laltu.sardar@iisertvm.ac.in, laltu.sardar.crypto@gmail.com
Course webpage: https://laltu-sardar.github.io/courses/26_crypto.html.