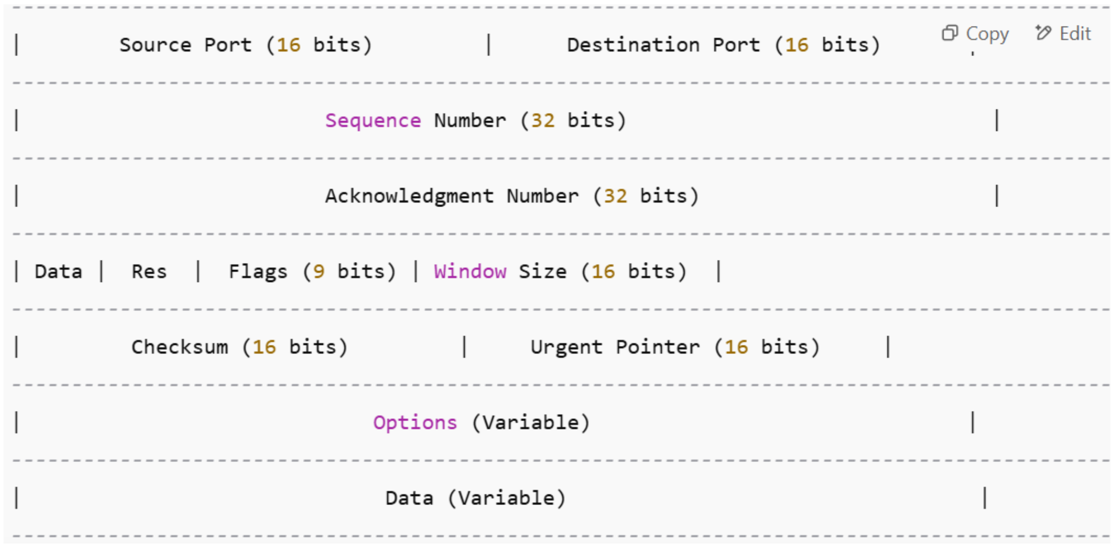
CHECKSUM AT DIFFERENET LAYERS

❖ CHECKSUM AT LAYER 3 (TRANSPORT LAYER):

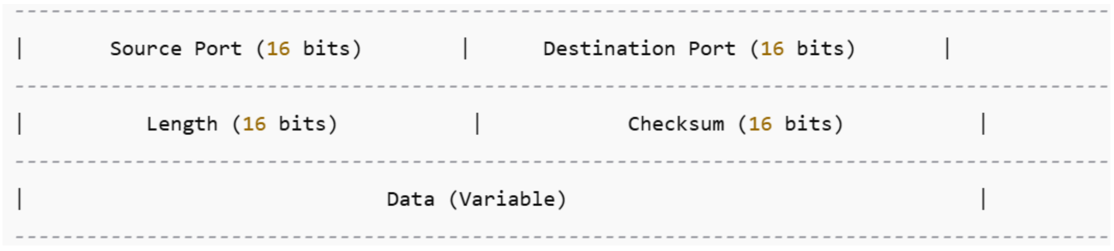
➢ TCP/IP LAYER :



```
-----------------------------------------------------------------------------------------
|        Source Port (16 bits)          |       Destination Port (16 bits)
-----------------------------------------------------------------------------------------
|                         Sequence Number (32 bits)                               |
-----------------------------------------------------------------------------------------
|                         Acknowledgment Number (32 bits)                         |
-----------------------------------------------------------------------------------------
| Data |  Res  |   Flags (9 bits) | Window Size (16 bits)  |
-----------------------------------------------------------------------------------------
|          Checksum (16 bits)          |      Urgent Pointer (16 bits)     |
-----------------------------------------------------------------------------------------
|                             Options (Variable)                              |
-----------------------------------------------------------------------------------------
|                              Data (Variable)                                 |
-----------------------------------------------------------------------------------------
```

- **Source Port (16 bits)** – Identifies the sending application.
- **Destination Port (16 bits)** – Identifies the receiving application.
- **Sequence Number (32 bits)** – Tracks the position of data in a stream.
- **Acknowledgment Number (32 bits)** – Confirms received data.
- **Data Offset (4 bits)** – Specifies header length.
- **Flags (9 bits)** – Control flags like SYN, ACK, FIN, RST, PSH, URG.
- **Window Size (16 bits)** – Flow control mechanism.
- **Checksum (16 bits)** – Error detection.
- **Urgent Pointer (16 bits)** – Priority data indication.
- **Options (Variable)** – Extra settings for performance or security.
- **Data (Variable)** – Actual payload from the application layer.

➢ UDP LAYER :



```
-----------------------------------------------------------------------------------------
|       Source Port (16 bits)          |       Destination Port (16 bits)       |
-----------------------------------------------------------------------------------------
|          Length (16 bits)             |            Checksum (16 bits)           |
-----------------------------------------------------------------------------------------
|                              Data (Variable)                                   |
-----------------------------------------------------------------------------------------
```

CHECKSUM AT DIFFERENET LAYERS

- **Source Port (16 bits)** – Identifies sending application.
- **Destination Port (16 bits)** – Identifies receiving application.
- **Length (16 bits)** – Total UDP segment size.
- **Checksum (16 bits)** – Ensures integrity.
- **Data (Variable)** – Payload from the application layer.

➢ How Transport layer is secure (TCP):

1. TLS :

When TLS is applied over TCP (HTTPS, Secure SMTP, etc.), the data encryption process happens **AFTER** the Transport Layer headers are added.

- The **application layer** sends data (e.g., an HTTP request).
- The **transport layer (TCP)** breaks the data into segments and adds headers.

```
---------------------------------
| TCP Header |  HTTP Data |
---------------------------------
```

- TLS takes the **TCP payload (HTTP data)** and encrypts it using **symmetric encryption (AES, ChaCha20, etc.)**.
- The encrypted payload is **wrapped inside a TLS record**.

```
---------------------------------
| TCP Header |  🔒 TLS Record (Encrypted Data) |
---------------------------------
```

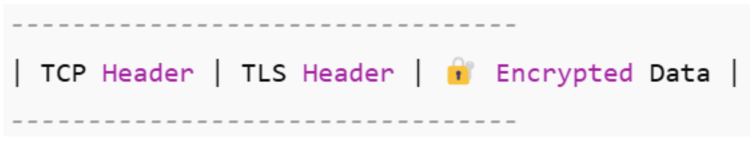
- A **TLS Record Header** is added to identify the type of encrypted data.
- This includes:

Content Type (e.g., "Application Data")

TLS Version

CHECKSUM AT DIFFERENET LAYERS

Length of Encrypted Data



```
---------------------------------
| TCP Header | TLS Header | 🔒 Encrypted Data |
---------------------------------
```

Option 2: Diffie-Hellman (DHE/ECDHE - Secure, Used in TLS 1.3)

1. **Client and server generate their own private keys.**
2. Each side **derives a public key** from their private key.
3. They **exchange public keys** and use them to compute a **shared session key.**
4. Since private keys are **never transmitted**, even if an attacker intercepts the public keys, they **cannot compute the session key**.

✔Secure → This provides Perfect Forward Secrecy (PFS), meaning past sessions remain safe even if a key is stolen later.

❖ Security at layer 3 :

At **Layer 3 (Network Layer)** of the TCP/IP model, **IPsec (Internet Protocol Security)** encrypts network packets to ensure **confidentiality, integrity, and authentication** of IP traffic. Here's a **detailed step-by-step breakdown** of how IP packets are encrypted at Layer 3 using **IPsec**.

IPsec operates at the Network Layer (Layer 3) and encrypts IP packets before transmission.
It works in **two modes**:

Transport Mode → Encrypts only the data (payload) inside the IP packet.

Tunnel Mode → Encrypts the entire IP packet (header + payload).

CHECKSUM AT DIFFERENET LAYERS

Key Components of IPsec: ✔**Encryption Algorithms:** AES (Advanced Encryption Standard), 3DES
✔**Authentication Algorithms:** HMAC-SHA-256, RSA Signatures
✔**Key Exchange Protocol:** IKE (Internet Key Exchange)

Step 1: Initiating a Secure Connection (IKE Phase 1)

Before data encryption starts, the two devices (client & server, or two routers) must establish a secure session.

✅ Process:
1 The sender and receiver exchange **IKE (Internet Key Exchange) messages**.
2 **Diffie-Hellman (DH) Key Exchange** is used to establish a **shared secret key**.
3 Both sides authenticate each other using **digital certificates** or **pre-shared keys**.
4 An **IPsec Security Association (SA)** is created to define the encryption and authentication parameters.

Once a secure connection is established, data packets are encrypted using **ESP (Encapsulating Security Payload)** or authenticated using **AH (Authentication Header)**.

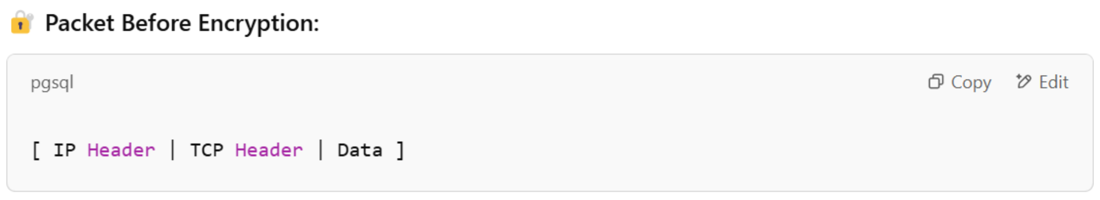
- **Transport Mode (Only Encrypts Payload)**

Original **IP header remains intact** for normal routing.

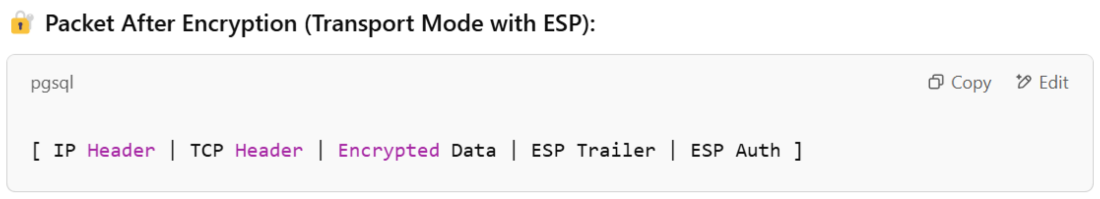
Only the **payload (TCP/UDP segment) is encrypted**.

Best for **end-to-end encryption** (e.g., between a laptop and a web server).

CHECKSUM AT DIFFERENET LAYERS


🔐 Packet Before Encryption:

```pgsql
[ IP Header | TCP Header | Data ]
```


🔐 Packet After Encryption (Transport Mode with ESP):

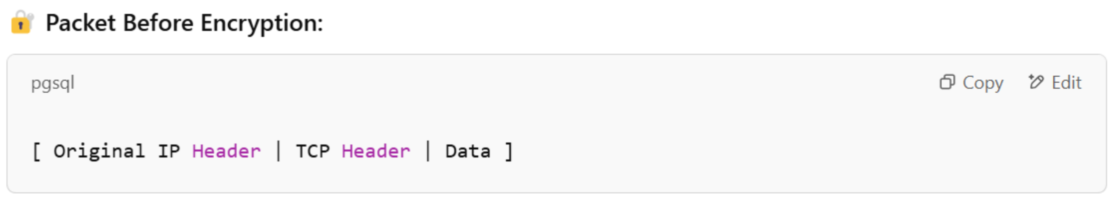
```pgsql
[ IP Header | TCP Header | Encrypted Data | ESP Trailer | ESP Auth ]
```

- Tunnel Mode (Encrypts Entire Packet)

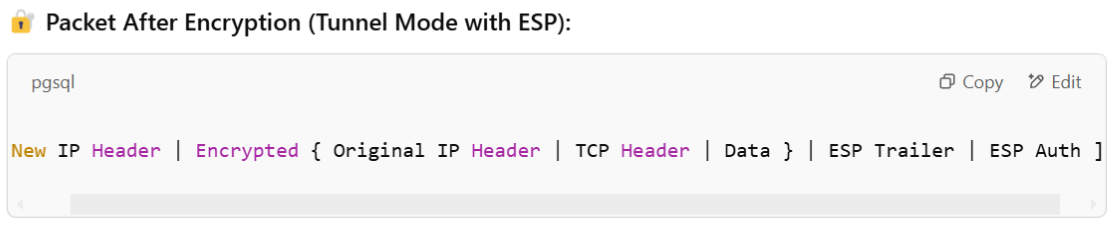
  Entire IP packet (header + payload) is encrypted.

  A **new IP header** is added to route the encrypted packet.

  Used for **VPNs (Virtual Private Networks)** to secure network traffic between gateways.


🔐 Packet Before Encryption:

```pgsql
[ Original IP Header | TCP Header | Data ]
```


🔐 Packet After Encryption (Tunnel Mode with ESP):

```pgsql
New IP Header | Encrypted { Original IP Header | TCP Header | Data } | ESP Trailer | ESP Auth ]
```

The encrypted **ESP packet** is sent over the network.

Since the data is encrypted, attackers **cannot see or modify the payload.**

Routers can still **route the packet** if Transport Mode is used.

CHECKSUM AT DIFFERENET LAYERS

✅ **At this stage, the encrypted packet is protected from:**
✔ **Eavesdropping (Confidentiality)** → AES encryption prevents packet sniffing.
✔ **Tampering (Integrity)** → HMAC ensures the data isn’t altered.
✔ **Spoofing (Authentication)** → Digital signatures verify sender authenticity.

1. What Does IPsec Do?

Encrypts IP packets so attackers cannot see the data.
Authenticates packets to ensure they are from a trusted source.
Protects against tampering using integrity checks.
Prevents replay attacks using sequence numbers.

Example:
Imagine Alice wants to send a **private letter** to Bob. Without encryption, anyone can **read or modify** the letter while it’s being delivered. IPsec acts like **a secure envelope**, ensuring:
✔ Only Bob can read it (**Encryption**).
✔ No one changes the message (**Integrity**).
✔ The message is really from Alice (**Authentication**).

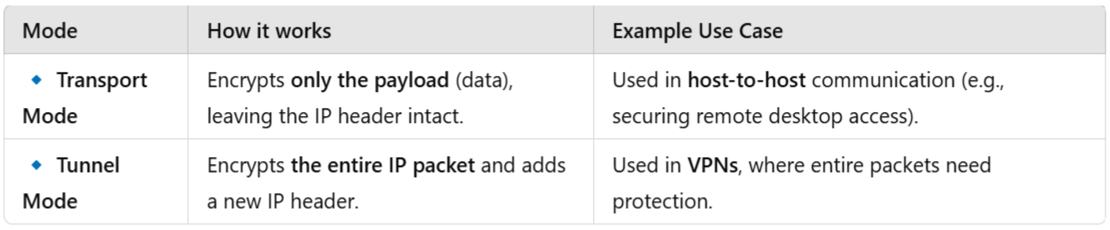
2. IPsec Components

IPsec consists of **three main building blocks**:

1 **ESP (Encapsulating Security Payload)** → Encrypts & authenticates packets ✅
2 **AH (Authentication Header)** → Only authenticates packets ❌ (No encryption)
3 **IKE (Internet Key Exchange)** → Securely exchanges encryption keys

3. Two Modes of IPsec

IPsec can work in two different **modes**, depending on how much of the packet it secures.



Mode	How it works	Example Use Case
◆ **Transport Mode**	Encrypts **only the payload** (data), leaving the IP header intact.	Used in **host-to-host** communication (e.g., securing remote desktop access).
◆ **Tunnel Mode**	Encrypts **the entire IP packet** and adds a new IP header.	Used in **VPNs**, where entire packets need protection.

ESP is most commonly used because it provides both **encryption** and **authentication**.
AH is rarely used because it does **not encrypt data**.

magine **Alice** is sending an email to **Bob**.

✅ **In Transport Mode** → The **content of the email is encrypted**, but the envelope (IP header) is visible.
✅ **In Tunnel Mode** → The **entire letter (email + envelope) is inside another envelope** (new IP header), making it completely hidden.

4. Step-by-Step Process of IPsec

Let’s break down **how IPsec encrypts and secures a packet** step by step.

Step 1: Key Exchange (IKE Phase 1)

Before encryption, both parties **agree on encryption keys** using **IKE (Internet Key Exchange)**.

Alice and Bob must exchange keys securely.
They use **Diffie-Hellman (DH) key exchange** to generate a shared secret key.

CHECKSUM AT DIFFERENET LAYERS

Example:
Imagine Alice and Bob want to share a **secret color** without others knowing. They mix their own **private colors** with a public color and exchange the result. Only they can recreate the **final secret color** (encryption key).

✅ This ensures that even if a hacker **eavesdrops**, they **cannot calculate the secret key**.

Step 2: Establish Security Association (SA)

A **Security Association (SA)** is created, which includes:

- **Encryption algorithm** (e.g., AES-256)
- **Authentication algorithm** (e.g., HMAC-SHA-256)
- **Key lifetime** (how long the encryption key is valid)

Example:
Think of an **SA** as a secret handshake between Alice and Bob, agreeing on **how they will encrypt and authenticate messages**.

Step 2: Establish Security Association (SA)

A **Security Association (SA)** is created, which includes:

- **Encryption algorithm** (e.g., AES-256)
- **Authentication algorithm** (e.g., HMAC-SHA-256)
- **Key lifetime** (how long the encryption key is valid)

Example:
Think of an **SA** as a secret handshake between Alice and Bob, agreeing on **how they will encrypt and authenticate messages**.

Step 3: Packet Encryption (ESP in Action)

Once keys are exchanged, IPsec **encrypts the data** using **ESP (Encapsulating Security Payload)**.

CHECKSUM AT DIFFERENET LAYERS

Example:
Alice sends **"Hello Bob"** → IPsec encrypts it into **"Xy9#%&8@!"**, making it unreadable to attackers.

Technical Process:

1 The **original packet** (IP Header + Data) is taken.
2 The **ESP header** is added.
3 The **payload (data) is encrypted** using AES-256.
4 A **message authentication code (MAC)** is added for integrity.

How It Looks Before & After Encryption:

```
Before IPsec:
------------------------------------------------------
| IP Header | Payload (Data: "Hello Bob") |
------------------------------------------------------

After IPsec (ESP Transport Mode):
--------------------------------------------------------------
| IP Header | ESP Header | Encrypted Payload | ESP Trailer |
--------------------------------------------------------------
```

The payload is **completely encrypted**, so hackers **cannot read it**.

If using Tunnel Mode: The **entire original packet is encrypted**, and a **new IP header** is added.

Step 4: Integrity Check & Authentication

To prevent **tampering**, IPsec uses **HMAC-SHA-256** to generate a **hash (fingerprint) of the packet**.

Example:
Bob receives Alice's encrypted message. He checks the **fingerprint** (HMAC) to verify:
✔The packet **came from Alice (authentication)**.
✔The message **was not altered (integrity check)**.

Step 5: Decryption & Packet Processing

When Bob receives the encrypted packet:
1 He **checks the integrity** (HMAC-SHA-256).
2 He **decrypts the payload** using the shared AES-256 key.
3 The **original message** is restored.

✅ **Final Output:** Bob successfully reads **"Hello Bob"**, and hackers see **random encrypted data**.

❖ Why ICMP should be encrypted ?

ICMP (Internet Control Message Protocol) messages are typically sent in plaintext, meaning they are not encrypted. This is because ICMP is primarily used for network diagnostics, error reporting, and operational information, such as:

- **Ping (Echo Request/Reply)** to check if a host is reachable.
- **Destination Unreachable** messages to inform about routing issues.
- **Time Exceeded** messages used in traceroute.

Should ICMP Be Encrypted?

Generally, **ICMP is not encrypted** because it is meant to be lightweight and quickly processed by networking devices. However, there are **security concerns** with plaintext ICMP:

1.

   Reconnaissance Attacks
   Attackers can use ICMP (e.g., ping sweeps) to map networks and identify active hosts.

2.

3.

ICMP Tunneling
ICMP packets can be used to exfiltrate data covertly (e.g., sending hidden payloads inside ICMP messages).

4.
5.

Denial of Service (DoS) Attacks
Large ICMP floods (e.g., Ping of Death, Smurf attacks) can disrupt network performance.

6.

When Should ICMP Be Encrypted?

- **Inside VPNs or Secure Networks**: If ICMP messages contain sensitive data (e.g., ICMP errors revealing internal IPs), encryption can be useful.
- **ICMP for Private Communication**: If you use ICMP-based tunneling or messaging, encryption can prevent data leaks.

Why Is ICMP Usually Not Encrypted?

1. **Overhead**: Encrypting ICMP adds processing overhead to networking devices.
2. **Limited Use Cases**: Most ICMP messages are for network debugging, not data transport.
3. **Compatibility**: Encryption could interfere with firewall rules and network diagnostics.

Alternative Security Measures:

- **Disable unnecessary ICMP types** (e.g., block external ping requests if not needed).
- **Rate-limit ICMP traffic** to prevent abuse.
- **Use VPNs or secure protocols (like IPsec) that encrypt all traffic, including ICMP.**

CHECKSUM AT DIFFERENET LAYERS

Conclusion:
ICMP encryption is usually **not necessary** for typical use cases, but if ICMP messages expose sensitive information, encrypting ICMP or using a VPN/IPsec can improve security.

❖ Calculation of Checksum after or before decryption ?

When verifying a checksum in encrypted communication, **it is more efficient to check the checksum after decrypting the payload** rather than before. Here’ s why:

1. Checking Before Decryption (Inefficient)

If you try to verify the checksum **before** decryption, the checksum will have been calculated on the **encrypted data**. This approach has several issues:

- The checksum will not match the original plaintext data, making it meaningless.
- If an attacker modifies the encrypted payload, the checksum may still be valid (depending on how it's computed), leading to potential security issues.
- Wasted computation: If the payload is corrupt, you will decrypt it anyway before realizing the error.

2. Checking After Decryption (Efficient)

Verifying the checksum **after** decryption is the correct approach because:

- The checksum is computed on the **original plaintext** data.
- If decryption fails, there’ s no need to waste time on checksum verification.
- This ensures the integrity of the **intended** data, not just the encrypted form.