HTTP & HTTPS: Complete Guide with Examples

The internet depends on protocols that allow computers, servers, browsers, and other devices to communicate with each other. Two of the most important protocols used when accessing websites are **HTTP (Hypertext Transfer Protocol)** and **HTTPS (Hypertext Transfer Protocol Secure)**.

Whenever you open a website such as a news site, online store, learning platform, or banking website, your browser communicates with a web server using HTTP or HTTPS.

In this guide, you'll learn what HTTP and HTTPS are, how they work, the differences between them, and why HTTPS is essential for modern websites.

What Is HTTP?

HTTP stands for Hypertext Transfer Protocol.

It is an application-layer protocol used to transfer resources between a **client** and a **web server**. These resources can include:

- HTML pages
- CSS files
- JavaScript files
- Images
- Videos
- JSON data
- Documents
- API responses

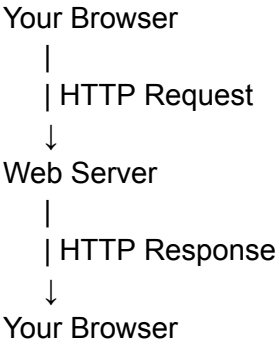
When you enter a website address into your browser, the browser acts as the **HTTP client**, while the website's server acts as the **HTTP server**.

For example:

http://example.com

The browser sends an HTTP request to the server, and the server returns an HTTP response.

Simple HTTP Communication



For example, when you request:

http://example.com/about

your browser may send a request similar to:

GET /about HTTP/1.1
Host: example.com

The server processes the request and sends back the requested webpage.

How HTTP Works

HTTP follows a **request-response model**.

Step 1: You Enter a URL

You type:

http://example.com

into your browser.

Step 2: Browser Finds the Server

The browser uses **DNS (Domain Name System)** to translate the domain name into an IP address.

For example:

example.com → 93.184.216.34

Step 3: Browser Sends an HTTP Request

The browser sends a request to the server.

For example:

GET / HTTP/1.1
Host: example.com

Step 4: Server Processes the Request

The web server receives the request and determines what resource should be returned.

Step 5: Server Sends an HTTP Response

The server responds with information such as:

HTTP/1.1 200 OK
Content-Type: text/html

followed by the webpage content.

Step 6: Browser Displays the Website

The browser interprets the HTML, CSS, JavaScript, images, and other resources and displays the webpage.

What Is HTTPS?

HTTPS stands for Hypertext Transfer Protocol Secure.

HTTPS is essentially HTTP communication protected by **TLS (Transport Layer Security)**.

A website using HTTPS normally has a URL beginning with:

https://

For example:

https://example.com

HTTPS provides important security properties, including:

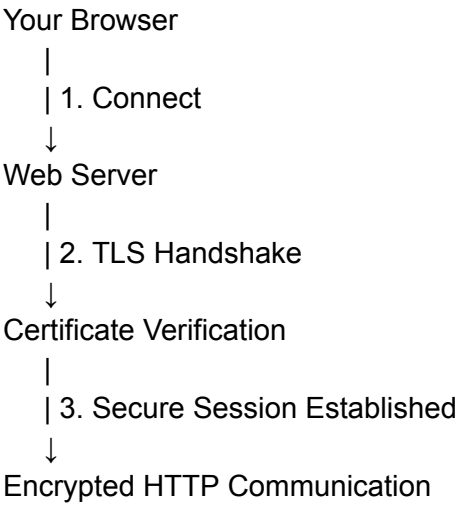
1. **Encryption**
2. **Authentication**
3. **Integrity**

These protections help prevent attackers from reading or secretly modifying data exchanged between your browser and the website.

How HTTPS Works

HTTPS uses **TLS** to establish a secure connection between your browser and the web server.

A simplified process looks like this:



Once the secure connection is established, HTTP messages are transmitted through the encrypted TLS connection.

The HTTPS TLS Handshake

The exact details depend on the TLS version and configuration, but conceptually the process works like this:

1. ClientHello

Your browser starts the TLS connection and provides information about supported TLS versions and cryptographic algorithms.

2. ServerHello

The server selects appropriate cryptographic parameters and provides its digital certificate.

3. Certificate Verification

The browser verifies that the certificate is trusted, valid for the requested domain, and otherwise acceptable according to certificate validation rules.

4. Key Establishment

The browser and server establish shared cryptographic key material.

5. Encrypted Communication

After the handshake, application data is protected using the negotiated encryption keys.

The result is that an attacker monitoring the network should not be able to read the protected HTTP content.

Example: HTTP vs HTTPS

Imagine you are ordering a product from an online store.

You enter your information:

Name: Ali
Address: Lahore
Card information: **********

Scenario 1: Using HTTP

Suppose the website uses:

http://shop-example.com

Your browser communicates with the server without TLS encryption.

If someone can successfully intercept the network traffic, sensitive information may be exposed or manipulated.

The problem is not simply that HTTP is "slow" or "old." The major issue is that **HTTP by itself does not provide TLS confidentiality or integrity protection**.

Scenario 2: Using HTTPS

Now suppose the website uses:

https://shop-example.com

Your browser establishes a TLS-secured connection with the website.

The information sent between the browser and server is protected against ordinary network interception.

Conceptually:

```
Browser
  |
  | Encrypted HTTPS traffic
  ↓
Internet
  |
  | Encrypted HTTPS traffic
  ↓
Web Server
```

An attacker monitoring the network should not be able to simply read your password, payment details, or other application data from the encrypted connection.

A More Familiar Example: Sending a Letter

Think of HTTP and HTTPS like sending a letter.

HTTP

Imagine putting a letter into a transparent envelope.

You → Transparent Envelope → Delivery Network → Receiver

Someone handling the envelope could potentially see what is inside.

HTTPS

Now imagine putting the letter into a securely sealed envelope.

You → Sealed Envelope → Delivery Network → Receiver

The delivery network can still carry the envelope, but it cannot simply read the letter inside.

This is a simplified analogy, but it helps explain the basic idea of encryption.

HTTP vs HTTPS

Feature	HTTP	HTTPS
Full form	Hypertext Transfer Protocol	Hypertext Transfer Protocol Secure
TLS encryption	No	Yes
Data confidentiality	Not provided by HTTP itself	Provided through TLS
Data integrity	Not provided by HTTP itself	Provided through TLS
Server authentication	No TLS authentication	TLS certificate authentication
Default port	80	443
URL	`http://`	`https://`
Suitable for sensitive data	No	Yes
Modern website standard	Generally no	Yes

HTTP Port 80

HTTP traditionally uses **TCP port 80**.

For example:

http://example.com:80

The `:80` is normally omitted because port 80 is the conventional default for HTTP.

HTTPS Port 443

HTTPS traditionally uses **TCP port 443**.

For example:

https://example.com:443

Again, the port number is normally omitted because 443 is the conventional default.

Note that modern HTTP can also run over different transport mechanisms, such as **HTTP/3 over QUIC**, so thinking of HTTPS exclusively as "HTTP over TCP 443" is an oversimplification.

What Is an SSL/TLS Certificate?

You may have heard people say:

> "This website has an SSL certificate."

Modern websites generally use **TLS**, not SSL. SSL is an older protocol that has been deprecated.

A TLS certificate is a **digital certificate** used to authenticate a website's identity and help establish a secure TLS connection.

For example:

https://itcodehub.org

When your browser connects, it can validate the website's certificate and establish a secure connection.

What Does the Padlock Mean?

Modern browsers may display a lock icon or other security indicator when a website uses HTTPS.

It generally indicates that the connection is protected by HTTPS/TLS.

However, **HTTPS does not mean that the website itself is trustworthy**.

For example, a malicious website can also obtain a valid TLS certificate.

Therefore:

HTTPS ≠ Automatically Safe Website

HTTPS primarily protects the **connection between your browser and the server**.

You should still check:

- Domain name
- Website reputation
- Content
- Download sources
- Login destination
- Payment information
- Signs of phishing

Three Major Security Benefits of HTTPS

1. Confidentiality

HTTPS encrypts application traffic so that ordinary network observers cannot simply read the transmitted content.

For example:

Password: MyPassword123

is not sent across the TLS-protected connection as readable plaintext.

2. Integrity

TLS helps detect unauthorized modification of protected data while it is being transmitted.

For example, an attacker should not be able to silently change:

Price: $100

into:

Price: $1

without the modification being detected by the cryptographic protections

3. Authentication

TLS certificates help your browser verify that it is communicating with the domain represented by the certificate.

This helps defend against certain impersonation and man-in-the-middle attacks.

What Is a Man-in-the-Middle Attack?

A **Man-in-the-Middle (MITM)** attack occurs when an attacker positions themselves between two communicating parties and attempts to intercept or manipulate their communication.

For example:

User
 |
 ↓
Attacker
 |
 ↓
Website

With properly configured HTTPS/TLS, the attacker should not be able to simply read or modify the protected application data.

Without encryption, intercepted HTTP traffic can be much easier to inspect and manipulate.

HTTP and HTTPS Example in a Public Wi-Fi Network

Imagine you are sitting in an airport or coffee shop and connect to public Wi-Fi.

You open:

http://example.com

An attacker on the same network may be able to observe or manipulate unencrypted traffic, depending on the network configuration and attack technique.

Now you visit:

https://example.com

The browser establishes a TLS-secured connection.

The network may still observe metadata such as the destination IP address and other connection information, but the contents of the HTTPS session are protected by encryption.

This is one reason HTTPS is especially important when using untrusted networks.

Does HTTPS Make a Website Completely Secure?

No.

HTTPS protects data while it is being transported between the client and server, but it does not automatically protect:

- A compromised server
- Weak passwords
- Vulnerable web applications
- SQL injection
- Cross-site scripting
- Malware
- Phishing
- Insecure APIs
- Poor access controls
- Stolen credentials

For example, if a website has a serious SQL injection vulnerability, simply enabling HTTPS will not fix that vulnerability.

Think of HTTPS as one important layer of a larger security architecture.

HTTP Redirect to HTTPS

Many websites automatically redirect HTTP requests to HTTPS.

For example:

http://example.com
↓
https://example.com

A server may return a redirect such as:

HTTP/1.1 301 Moved Permanently
Location: https://example.com/

The browser then requests the HTTPS version.

Websites can also use **HSTS (HTTP Strict Transport Security)** to instruct compatible browsers to use HTTPS for future connections.

HTTPS and SEO

HTTPS is also important for website owners and SEO.

Google has used HTTPS as a **ranking signal**, although it is only one factor among many.

More importantly, HTTPS provides a safer experience for visitors and is expected across the modern web.

HTTPS is particularly important for websites that handle:

- Login credentials
- Contact forms
- Personal information
- Payments
- User accounts
- Educational platforms
- E-commerce transactions

For a professional website, HTTPS should be considered a basic security requirement rather than an optional feature.

How to Check Whether a Website Uses HTTPS

Look at the website address in your browser.

HTTP:

http://example.com

HTTPS:

https://example.com

You can also click the browser's security indicator to inspect connection and certificate information, although the exact interface varies by browser.

HTTP Status Codes

HTTP also uses **status codes** to communicate the result of a request.

Common examples include:

200 OK

The request was successful.

200 OK

301 Moved Permanently

The requested resource has been permanently redirected.

301 Moved Permanently

404 Not Found

The requested resource could not be found.

404 Not Found

403 Forbidden

The server understood the request but refuses to authorize it.

403 Forbidden

500 Internal Server Error

The server encountered an unexpected problem.

500 Internal Server Error

These status codes can be used with both HTTP and HTTPS; HTTPS changes how the HTTP exchange is transported and protected, not the meaning of the HTTP status codes.

HTTP Methods

HTTP provides several request methods.

GET

Used to retrieve data.

GET /articles HTTP/1.1

POST

Commonly used to submit data.

POST /login HTTP/1.1

PUT

Often used to replace or update a resource.

PATCH

Often used to partially modify a resource.

DELETE

Used to request deletion of a resource.

These methods are fundamental to web applications and REST-style APIs.

Example: Online Banking

Consider an online banking application.

You open:

https://bank-example.com

You enter:

Username: Ali123
Password: ********

Your browser sends the request through a TLS-protected HTTPS connection.

The server authenticates you and returns your account information through the same protected connection.

When you transfer money, the request is also protected in transit.

Conceptually:

Your Browser
    |
    | 🔐 HTTPS/TLS
    ↓
Bank Server
    |
    ↓
Database

The HTTPS layer helps protect communication between your browser and the bank's web server.

The bank still needs additional security controls such as authentication, authorization, fraud detection, secure coding, database security, and transaction monitoring.

HTTP vs HTTPS: Which One Should You Use?

For modern websites, **HTTPS should be the default choice**.

Use HTTPS when your website:

- Has a login system
- Collects user information
- Processes payments
- Contains forms
- Uses APIs
- Stores user accounts

- Runs an e-commerce store
- Publishes educational content
- Uses cookies or authentication sessions

Even a simple informational website should normally use HTTPS.

Key Differences in Simple Words

The easiest way to remember the difference is:

HTTP
↓
Communication without TLS protection

HTTPS
↓
HTTP + TLS security

Or:

HTTP = Web communication
HTTPS = Web communication + encrypted/authenticated TLS connection

Conclusion

HTTP and HTTPS are fundamental technologies of the web.

HTTP defines how browsers and servers exchange web requests and responses. HTTPS adds **TLS-based security**, helping provide confidentiality, integrity, and authentication for that communication.

A simple HTTP connection can expose users to greater risks when sensitive information is transmitted over an untrusted network. HTTPS protects the connection and is therefore the standard choice for modern websites.

The real-life example of online shopping or online banking makes the difference easy to understand: **HTTPS acts like a secure communication channel between your browser and the website, while HTTP does not provide that TLS protection.**

For website owners, developers, and cybersecurity learners, understanding HTTP and HTTPS is essential because these protocols form the foundation of modern web communication.