Before mid.

Web Application Architecture

* What is an Architecture?

=D There is no unique definition of the term "Architecture".

* Components of web architecture:

=D Client: Generally a browser (user agent) is controlled by a user to operate the Web application. The clients ~~functionality~~ functionality can be expanded by installing plug-ins and applets.

=D Firewall: A piece of software regulating the communication between insecure networks (the internet) and secure networks (eg., corporate LANs). This communication is filtered by access rules.

=D Proxy: A proxy is typically used to temporarily store Web pages in a cache. However, proxies can also assume other functionalities, e.g., adapting the contents for users (customization) or user tracking.

=D Web servers: A Web server is a piece of software that supports various Web protocols like HTTP, HTTPS, etc., to process client requests.

=D Database Servers: This server normally supplies an organization's production data in structured form, eg., in tables. (mysql)

=D Media servers: This component is primarily used for content streaming of non-structured bulk data (e.g., audio or video).

=D Content management servers: Similar to a database server, a content management server holds contents to serve an application. These contents are normally available in the form of semi-structured data, eg., XML documents.

=D Application Servers: An application server holds the functionally required by several applications, e.g., workflows or customization.

=D Legacy application: A ~~legacy~~ legacy application is an older system that should be integrated as an internal or external component.

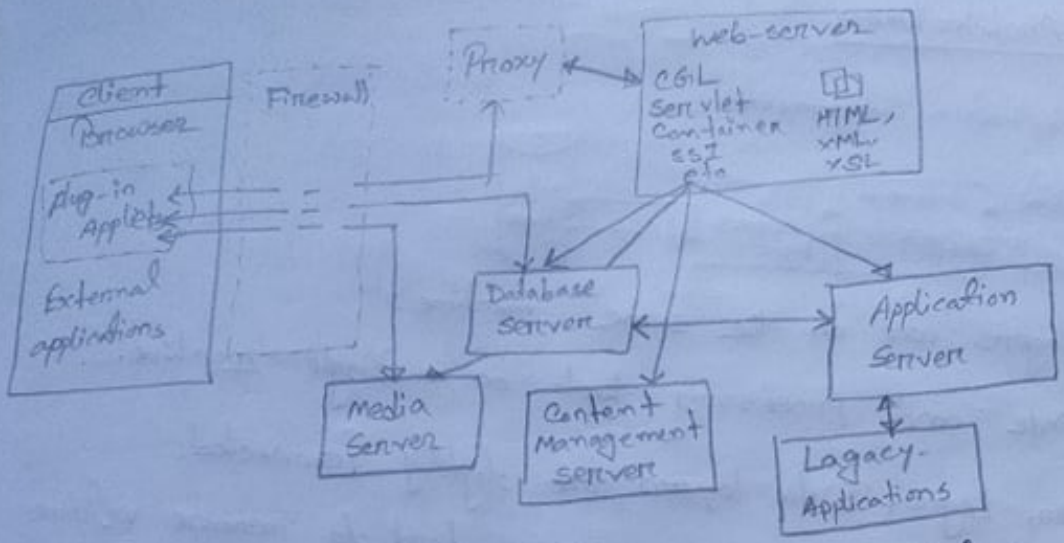


Figure: Basic components of web application architectures

N Tier architectures:

N-tier architectures have the same components

⇒ Presentation.

⇒ Business / Logic

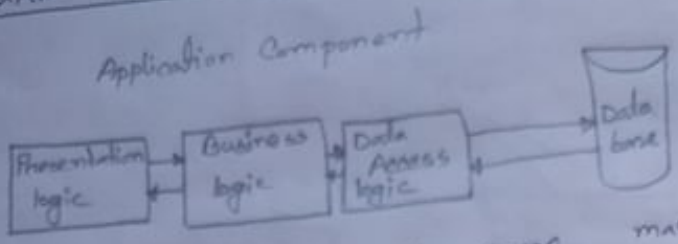
⇒ Data

N-tier architectures try to separate the components into different tiers / layers

⇒ Tier: Physical separation [server সবগুলো client আলাদা]

⇒ Layer: logical separation

1-tier Architecture:



All 3 layers are on the same machine.
⇒ All code and processing kept on a single machine.

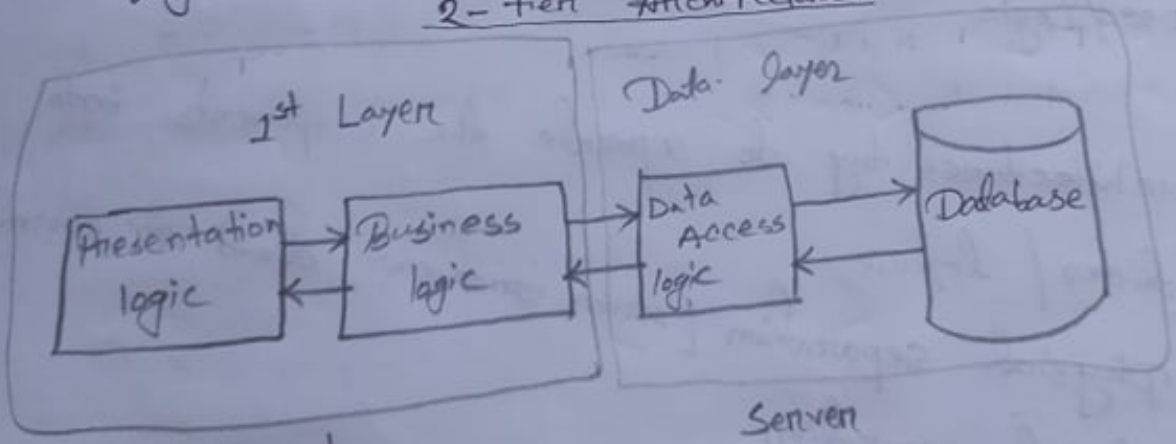
Presentation, Logic, Data layers are tightly connected.

⇒ Scalability: Single processor means hard to increase volume of processing.

⇒ Portability: Moving to a new machine may mean rewriting everything.

⇒ Maintenance: Changing one layer requires changing other layers.

2-tier Architecture:



Client

Server

↓

সার্ভার computer.

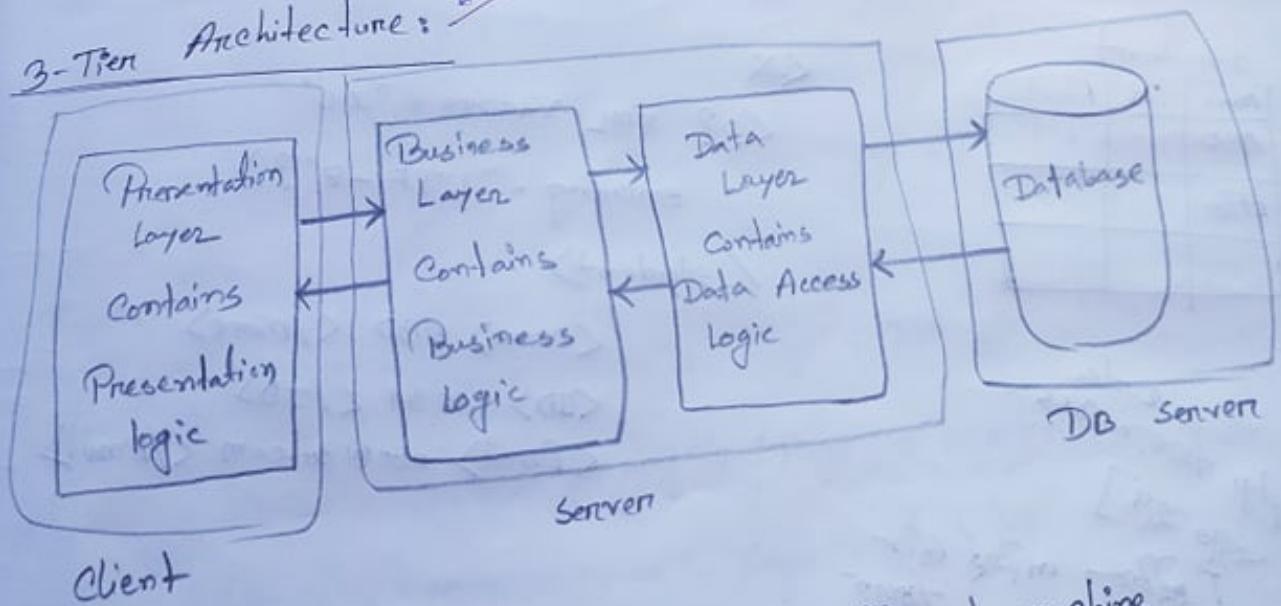
Database runs on Server:

- Separated from client
- Easy to switch to a different database

Presentation and Logic layers still tightly connected.

- Heavy load on server.
- Potential congestion on network
- Presentation still tied to business logic.

3-Tier Architecture:



- Each layer can potentially run on a different machine
- Presentation, logic, data layers disconnected.

Web Technologies — Sunday

HTML
CSS
JS
XML ← Extensible Markup language
JSON ← Javascript object Notation [Data pass]
AJAX ← Asynchronous Javascript and XML
REST ← Representational state Transfer
HTTP

XML

Student

Name	Id	Email
ABD	..	...
EFG	..	...

Ex

```
<? XML version = "1.0"
    encoding = "UTF-8" ?>
<student>
    <Name> ABD </Name>
    <ID> 1234 </ID>
    <Email> abc@dcc.com </Email>
</student>
```

[php থেকে Data pass করতে হবে]

[এগুলো অবশ্যই উল্টো লিখতে হবে]

```
< → &lt;
< → &gt;
& → &amp;
" → &quot;
' → &apos;
```

```
<student>
    <student>
        <Name> ..... </Name>
    </student>
</student>
```

01616106868

JSON

For a single column :-

```
{"Name": "ABCD", "ID": 123, "Email": "abc@def.com"}
```

For Entire table :

```
{"student": [
        {"Name": "ABCD"...},
        {. . . . . . },
        {. . . . . . },
        { . . . . .
    ]
}
```

[প্রতিটা JSON এ Array থাকবে]

[XML & JSON পার্থক্য]

→ book এর আন্ডারে JSON file তৈরি করা।

→ Library → 4 book.

Book

Name	Page
Data	305
Machine	550

JSON

```
{"Book": [
        {"name": "Data structure", "page": 305},
        {"name": "Machine Larning", "page": 550},
        {"Name": "Microprocessor", "page": 350},
        {"Name":
        {
    ]
}
```

XML

```
<Book>
    <Name> Data structure </Name>
    <Page> 305 </Page>
    <Name> Machine Learning </Name>
    <Page> 550 </Page>
    <Name> Microprocessor </Name>
    <Page> 350 </Page>
</Book>
```

after star * marked page

typical 3-tier Architecture:

Architecture Principles:

- Client-Server architecture.
- Each tier (Presentation, Logic, Data) should be independent and should not expose dependencies related to the implementation
- Unconnected tiers should not communicate.
- Change in a platform affects only the layer running on that particular platform.

Presentation Layer:

- Provides user interface.
- Handles the interface with the user.
- Sometimes called the GUI or client view or front-end.
- Should not

Logic Layer:

- The set of rules for processing information.
- Can accommodate many users.
- Sometimes called middleware / back-end.
- Should not contain presentation or data access code.

Data Layer:

=D The physical storage layer for data persistence.
=D Manages access to DB or file system.
=D Sometimes called back-end.
=D Should not contain Presentation or business logic code.

The 3-tier Architecture for web Apps:

Presentation Layer:

static or dynamically generated content rendered by the browser (front-end).

Logic Layer:

A dynamic content processing and generation level application server, e.g., Java EE, ASP.NET, PHP, ColdFusion platform (middleware).

Data Layer:

A database, comprising both data sets and the database management system or RDBMS software that manages and Provides access to the data (back-end).

3-Tier Architecture - Advantages:

Independence of layers:

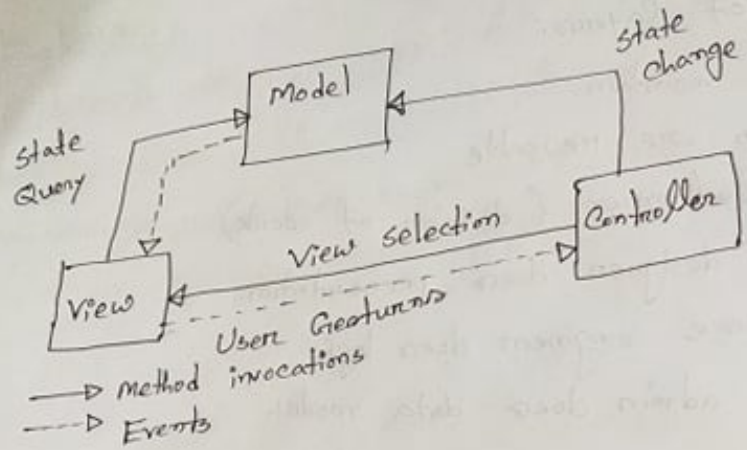
- Easier to maintain
- Components are reusable
- Faster development (division of work)
  - Web designer does presentation
  - Software engineer does logic
  - DB admin does data model.

MVC (Model-View-Controller)

The model-view-controller Pattern:

Design pattern for graphical systems that promotes separation between model and view.

With this pattern the logic required for data maintenance (database, text file) is separated from how the data is viewed (graph, numerical) and how the data can be interacted with (GUI, command line, touch)



Model :

=D Manages the behavior and data of the application domain.

=D responds to requests for information about its state (usually from the view)

=D follows instructions to change state (usually from the controller)

View :

=D renders the model into a form suitable for interaction, typically a user interface (multiple views can exist for a single model for different purposes)

Controller :

⇒ receives user input and initiates a response by making calls on model objects.

⇒ accepts input from the user and instructs the model and viewpoint viewpoint to perform actions based on that input.

The MVC Pattern (in Practice)

Model :

⇒ Contains domain-specific knowledge.

⇒ Records the state of the application.

→ E.g., what items are in a shopping cart

⇒ Often linked to a database

⇒ Independent of view.

→ One model can link to different views.

2. # View :

⇒ Presents data to the user

⇒ Allows user interface

⇒ Does no processing

Controllers:

=D defines how user interface reacts to user input (events)

=D receives messages from view (where events come from)

=D Sends messages to model (tells what data to display)

The MVC for Web Applications:

Model:

=D Database tables (persistent data)

=D session information (current system state data)

=D rules governing transactions

Views:

=D (X) HTML

=D CSS style sheets

=D Server-side templates

Controllers:

=D client-side scripting

=D http request processing

=D business logic / preprocessing

3-tier Architecture Vs. MVC Architecture:

Communication:

3-tier: The presentation layer never communicates directly with the data layer - only through the logic (linear - topology).

MVC: All layers communicate directly (triangle topology)

Usage:

3-tier: Mainly used in web applications where the client, middleware and data tier run on physically sp separate platforms.

MVC: Historically used on applications that run on a single graphical workstation (applied to separate platforms as Model 2).

Framework Library :

Node JS
MVC

class / id HTML/ CSS Design

we provide code
in a way the
framework accepts.

Hollywood principle → You will not call, we will call you.

socket for HTTP server

15/12/2024

Hyper Text Transfer Protocol (HTTP)

(FTB) → File Transfer করার জন্য।

Web and HTTP : Some Terms

* HTTP stands for "Hyper Text Transfer Protocol".
* A web page consists of many objects.
* Object can be HTML file, JPEG image, video stream chunk, audio file, ...
* Web page consists of base HTML-file which includes several referenced objects.
* Each objects is addressable by a uniform resource locator (URL) [Uniform resource identifier]

* Example URL :

www.cs.rutgers.edu/~sn624/index.html

Host name — path name

HTTP Overview :

HTTP : hypertext transfer protocol

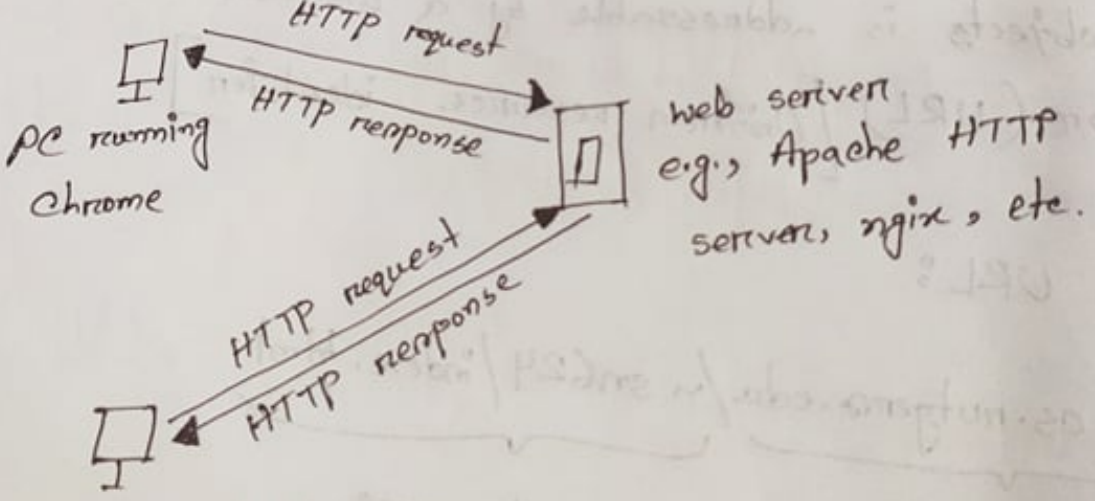
* client/server model.

client : browser that requests, receives, "displays" web projects.

Server : Web server sends objects in response to requests.

Versions :

* HTTP 1.0 : RFC 1945
* HTTP 1.1 : RFC 2068



HTTP method types:

* GET:
  * Get the resource specified in the requested URL. URL may refer to a data handling process.

* TRACE:
  * Trace Proxies

* Option:
  * Determine servers capability

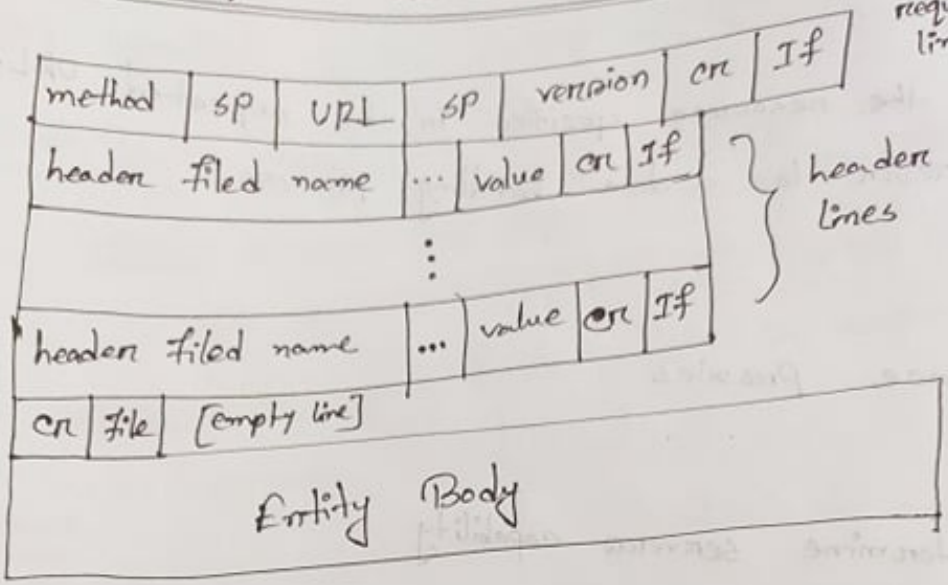
* Post:
  * send entities (specified in the entity body) to a data-handling process at the requested URL.

* HEAD:
  * Asks server to leave requested objects out of response, but send the rest of the response.

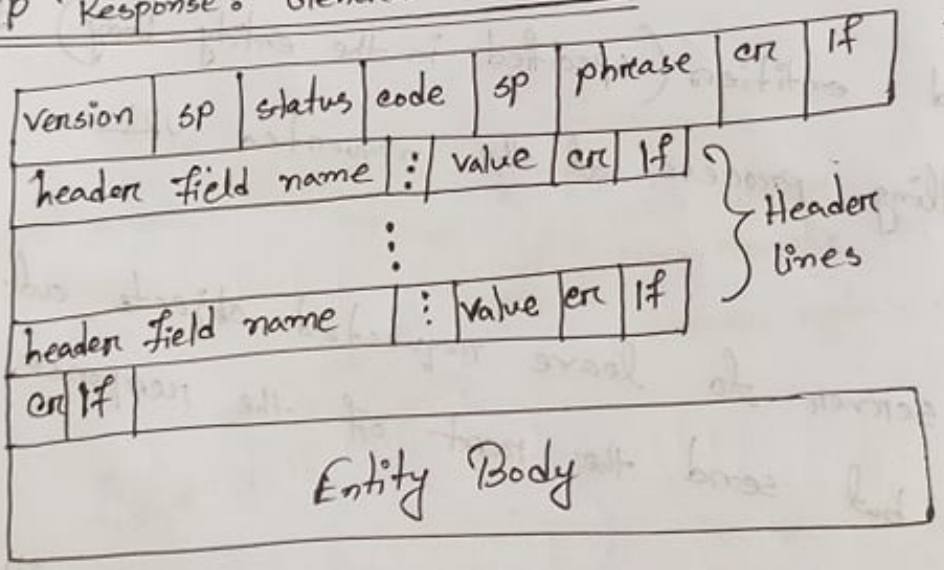
* PUT:
  * Update a resource at the requested URL with the new entity specified in the entity body.

* DELETE:
  * Deletes file specified in the URL.

HTTP Request: General format:-



HTTP Response: General format:-



Unlike HTTP request, No method name.

HTTP message : response message

HTTP/1.1 200 OK ← status line (Protocol status code status phrase)

response header lines →
- Connection : close
- Date : Thu 06 Aug 1998 12:00:15 GMT
- Server : Apache/1.3.0 (Unix)
- Last-modified : Mon, 22 Jun 1998 .....
- Content-Length : 6821
- Content-Type : text/html

data data data data .....

↑ data, e.g., requested HTML file in entity body.

HTTP : Request

GET (method) / index.html (URL) HTTP/1.1 (Protocol Version)

Headers
- Host : www.example.com
- User-Agent : Mozilla/5.0
- Accept : text/html, */*
- Accept-Language : en-us
- Accept-charset : ISO-8859-1, utf-8
- Connection : keep-alive

Blank line

Body (optional) {

HTTP Response :

Version ↑ — Status ↑ — Status message ↓

HTTP/1.1 200 OK

Headers:
- Data: Thu, 24 Jul 2008 17:36:27 GMT
- Server: Apache-coyote/1.1
- Content-Type: text/html; charset = UTF-8
- Content-Length: 1896

blank line

body:
- <html>
- ....
- </html>

Uploading form input : GET and Post

POST method :

- Web page often includes form input
- Input is uploaded to server in entity body.

GET

Posted content not visible in the URL

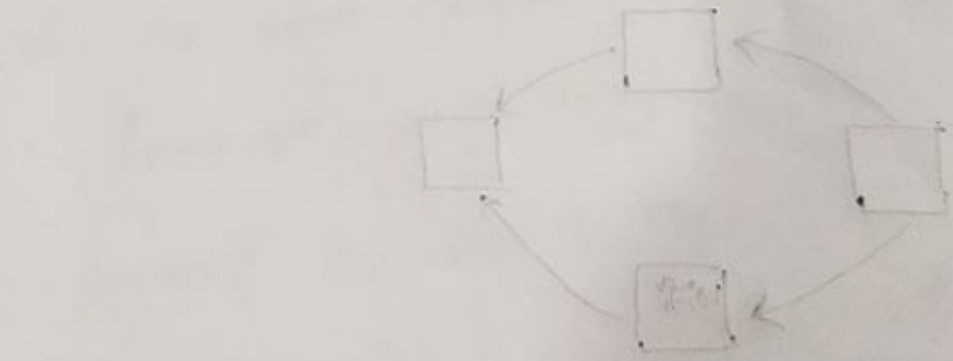
- Free form content (ex: images) can be posted since entity body interpreted as data bytes.

GET method:

- Entity body is empty.
- Input is uploaded in URL field of request line.

Example:

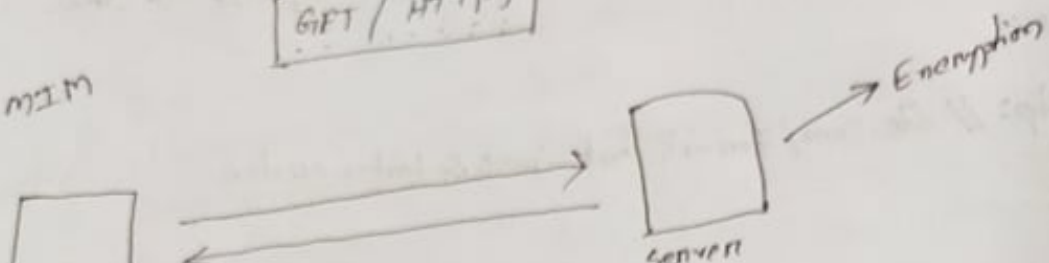
http://site.com/form?first=jane&last=austen



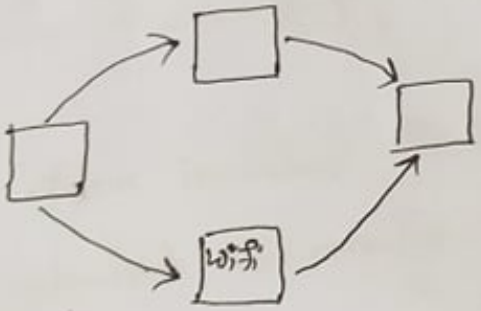
offline
Sunday.

GET / HTTP1

MITM

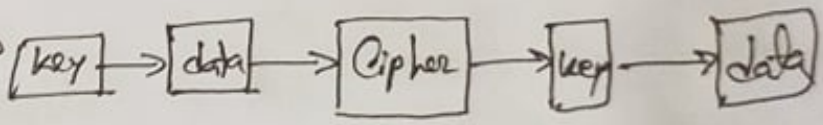


HTTP



Symmetric

Asymmetric

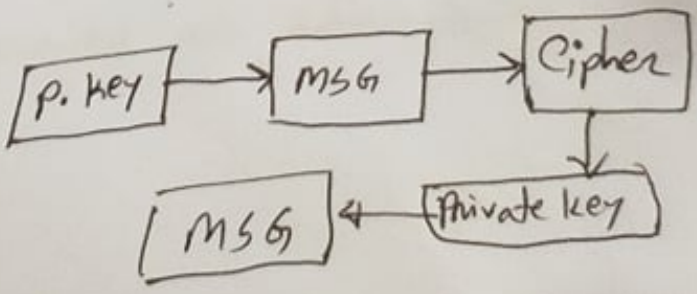


↓ Generate
1 way mathematical function

Key pair

Public key

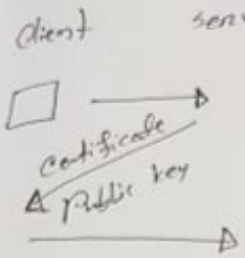
Private key



SSL Handshake / TLS Handshake

How Encryption works with server or HTTP :

1. Browser Request a Connection
2. Server sends certificate (Public key)
3. Client Create a Server Session key
4. Encrypt the seam session key and send to server. [Encrypt = public key]
5. Server decrypt the session key.
3. Asymmetric encryption ends off and Related by Symmetric connection.
. Continue using this symmetric connection.



SSL Handshake / TLS Handshake

How Encryption works with server on HTTP :

1. Browser Request a Connection
2. Server sends certificate (Public key)
3. Client Create a Server Session key
4. Encrypt the session key and send to server. [Encrypt = public key]
5. Server decrypt the session key.
6. Asymmetric encryption ends off and Related by Symmetric connection.
7. Continue using this Symmetric connection.

RSA → Rivest- Shamir- Adleman

↓

Private, Public
key ဖြစ်တယ်၊

SSL / TLS

Summary :-