Intro to the Internet/HTML

CS472 Web Application Programming

Maharishi International University

Department of Computer Science

Associate Professor Asaad Saad

Maharishi International University - Fairfield, Iowa

Wholeness Statement

In this lecture we introduce the basic technologies that make up the Internet, the World Wide Web, and the Hyper Text Markup Language (HTML). We will see that many technologies are built on top of other technologies.

Life is found in layers and the TM Technique gives us access to the full range of our awareness and thoughts.

Client vs. Server-Side Applications

- Developers write applications with code, using a programming language.
- When code runs in the browsers it's called client-side application, and when runs on a different machine it's called server-side application or cloud-based application.
- Browsers can parse HTML, CSS, and execute JavaScript. Users can inspect all code in the browser. Client-side applications are mostly for UI.
- Servers can run any programming language: JavaScript (Node), TypeScript, Python, Java, PHP, C#.. etc. Server-side applications are mostly for secure and heavy processes.

Application Data

- Client-Side applications may save data in the browser (which can be accessible by users, not secure), usually UI state.
- Server-Side applications may save data on the server but that does not scale well (filesystem, or sessions in memory) or save data using another DBMS system that allows scalability.

The Internet

A connection of computer networks using the Internet Protocol (IP)

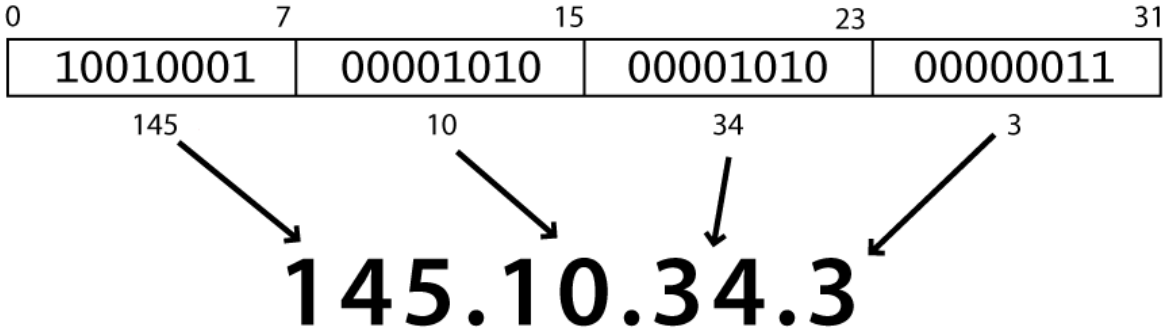
layers of communication protocols:

1. IP
2. TCP/UDP
3. HTTP/FTP/POP/SMTP/SSH…

Internet Protocol (IP)

A simple protocol for attempting to send data between two computers

Each device has a 32-bit IP address written as four 8-bit numbers (0-255)
There are two types of IP addresses, servers used to have **static** IP address while users usually get a **dynamic** IP address from their ISP.



IPv6 addresses are 128-bit IP address written in hexadecimal and separated by colons. An example IPv6 address could be written like this: 3ffe:1900:4545:3:200:f8ff:fe21:67cf

Transmission Control Protocol (TCP)

TCP (Transmission Control Protocol) is a standard that defines how to establish and maintain a network conversation via which application programs can exchange data. TCP works with the Internet Protocol (IP).

Multiplexing

Multiple programs using the same IP address, by using a **port**, a number given to each program or service

- port 80: default
- port 443 for secure connection
- port 21: ftp
- port 22: ssh

Domain Name System (DNS)

DNS servers map written names to IP addresses

www.cs.miu.edu → 69.18.50.54

Many systems maintain a local DNS cache called a **`host`** file:

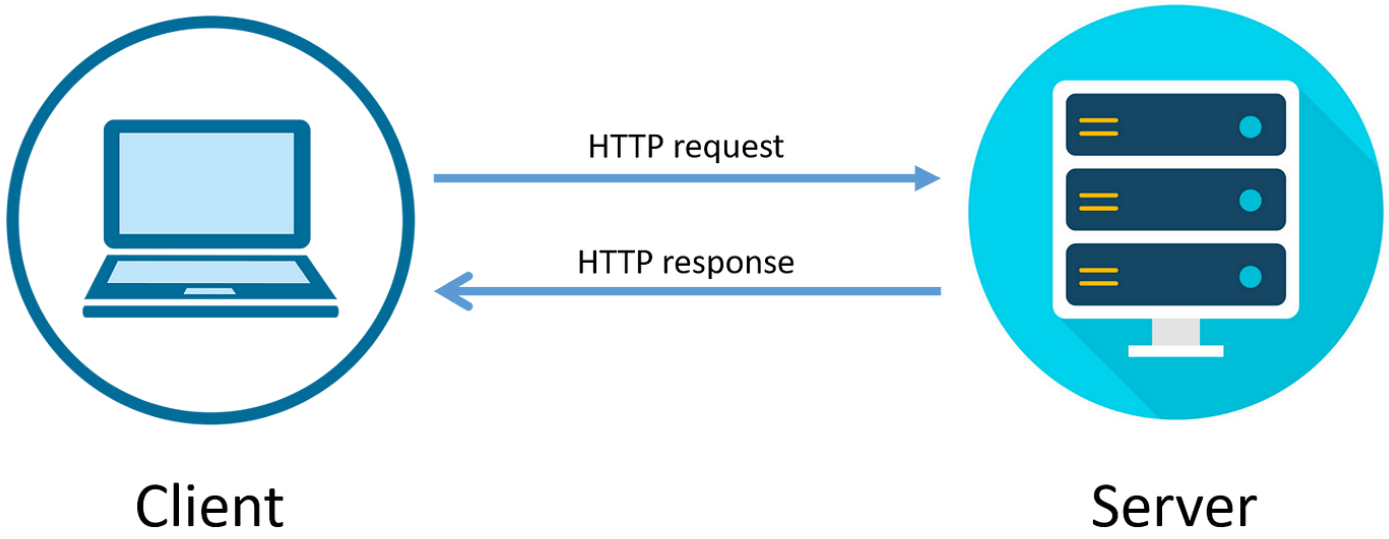
- Windows: `C:\Windows\system32\drivers\etc\hosts`
- Mac: `/etc/hosts`
- Linux: `/etc/hosts`

Hypertext Transport Protocol (HTTP)

HTTP is the underlying protocol used by the World Wide Web and this protocol defines **how messages are formatted**, and what actions Web servers and clients should take in response to various **commands**.

HTTP Request

1. The client sends an **HTTP** request to the server
2. The server sends a response back



Example

Request

Header

```
GET /index.html HTTP/1.1
Host: www.miu.edu
User-Agent: Mozilla/5.0
Connection: keep-alive
Accept: text/html
If-None-Match: fd87e6789
```

Body

```
No body with GET requests
```

Response

Header

```
HTTP/1.1 200 OK
Content-Length: 16824
Server: Apache
Content-Type: text/html
Date: Monday 8 Dec 2020
Etag: h64w175
```

Body

```
<!DOCTYPE html>
<html>
...
```

***Demo**: Inspect req/res in Chrome devtools, and Rest Client.*

HTTP Verbs

- **GET** Retrieves data from the server
- **HEAD** Same as GET, but response comes without the body
- **POST** Submits data to the server
- **PUT** Replace data on the server
- **PATCH** Partially update a certain data on the server
- **DELETE** Delete data from the server
- **OPTIONS** Handshaking and retrieves the capabilities of the server

Encryption vs. Hashing

Encryption is a two-way function, what is encrypted can be decrypted with the proper key. Usually a key and salt are used.

Example: base64 (atob, btoa), RSA, Crypto, AES

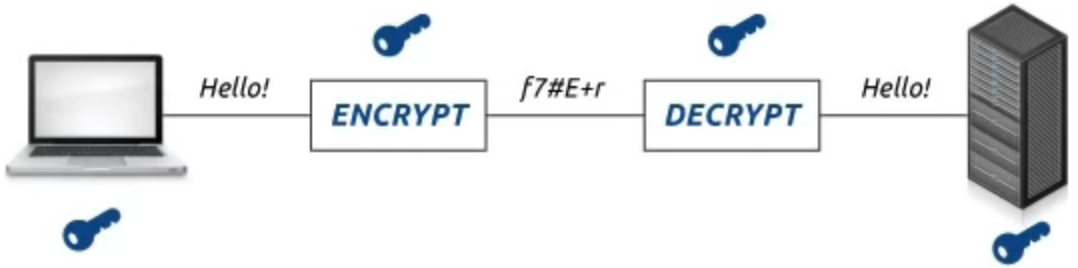
Hashing is a one-way function that scrambles plain text to produce a unique message digest (most of time with a fixed size). There is no way to reverse the hashing process to reveal the original plain text.

Example: MD5, SHA1/256/512, PBKDF2, BCrypt

How do we save passwords in your DB server?

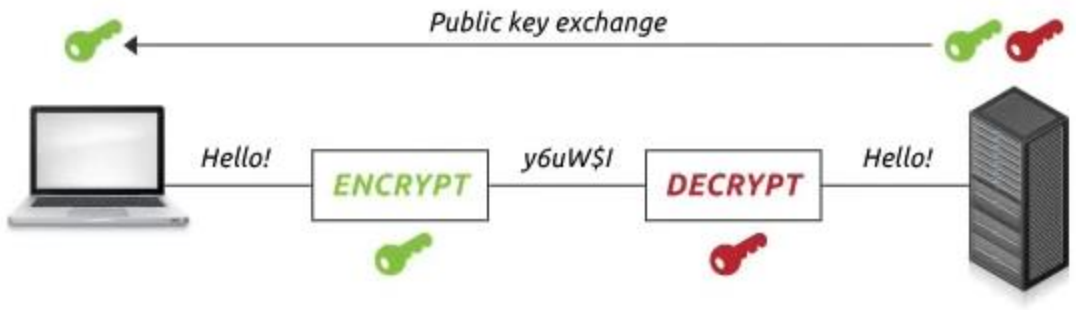
Symmetric Encryption

Symmetric cryptography is a type of encryption where the same key (secret key) is used to both encrypt and decrypt the data. Usually it's very fast because it uses 256 bits key.



Asymmetric Encryption

Asymmetric cryptography (public key cryptography), uses public and private keys to encrypt and decrypt data. **Public key** is used to encrypt the data, and the **Private key** is used to decrypt the data. This is more secure because it uses a 2048 key, but it is slow.



TLS (Transport Layer Security)

A standard security technology for establishing an encrypted connection between a web server and a client. This connection ensures that all data passed between the web server and the client remains private.

Digital Certificate

A certificate is an electronic credentials used to assert the online identities of individuals, applications and other entities on a network.
They are similar to an ID card (passport, driving license, diploma). A certificate is signed by (Certificate Authority CA).

A certificate is a file contains:

- Identity of owner
- Public Key of the owner
- Signature of the certificate issuer
- Expiration date

A signature is always made with the private key of issuer and can be validated with the public key of the issuer.

Certificate Issuer

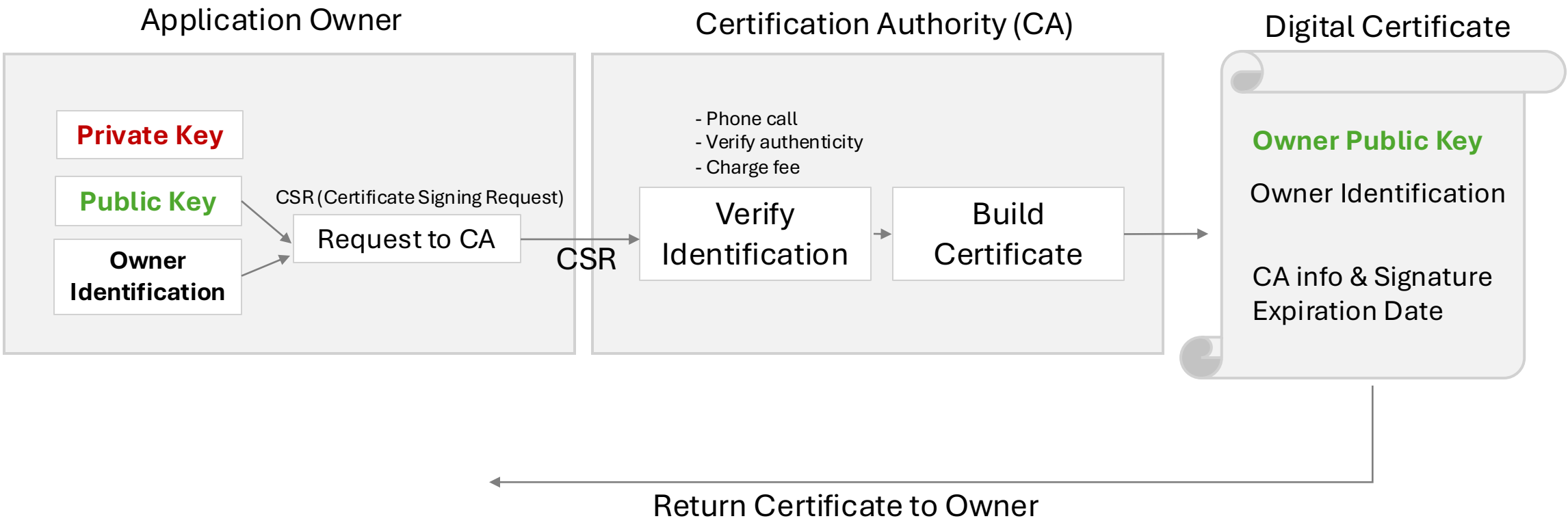
A certificate authority (CA) validates the identity of the certificate holder.

There are two types of CA:

- **RCA**: Trusted Root Certification Authorities (CA)
- **ICA**: Intermediate Certification Authorities (They own certificates from CA) (a security layer so private key for RCA is inaccessible)

All Root CA are pre-defined in the browser, so any client can validate the certificate (like a hotline between the browser and the issuer).

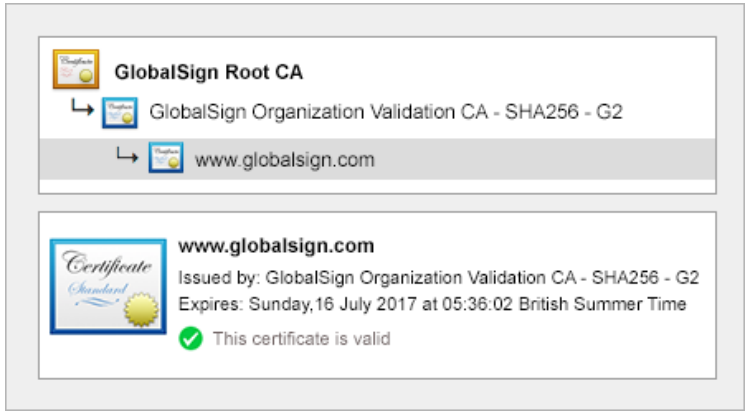
How Digital Certificates Are Created?



Using Digital Certificates

When a client contacts an application owner via TLS, the first thing they receive is the digital certificate. Clients check the validity of this certificate (if it was issued to the claimed app owner, not expired, the CA signature).

Usually a **chain of certificates** are sent to the client (the owner certificate signed by ICA, and ICA certificate signed by Root CA), since the client has the public key for all Root CAs, it validates the ICA certificate first and makes sure it is a certified entity, and then use its public key in that certificate to validate the owner certificate.



TLS Handshake

- **Client**: I want to connect to https://miu.edu
- **Server**: Sure, he's my certificate containing my asymmetric public key. It was signed by Comodo CA.
- **Client**: validates the signature of the certificate using the public key of trusted issuers (Comodo CA), if valid, it creates a new symmetric secret key and encrypts it with the asymmetric public key of MIU in the certificate.
- **Server:** decrypts the asymmetric public key with its asymmetric private key to get the symmetric secret key of the client.

All future communication is encrypted and decrypted with the symmetric secret key.

If the client were to connect to the same server the next day, a **new secret key** would be created.

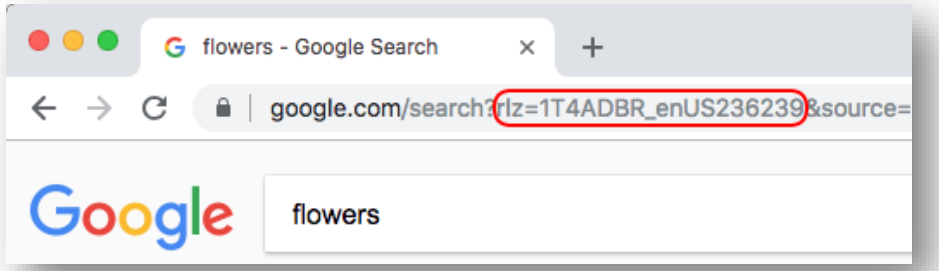
Web Communication In a Nutshell

We generally use the **asymmetric system** to exchange the secret key and then **symmetric system** to encrypt and decrypt messages.

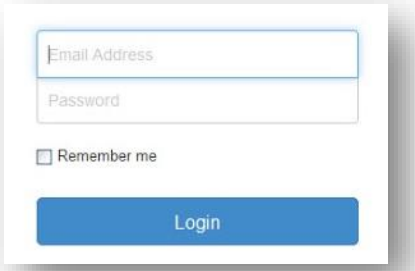
- The server sends its public key to the client
- The client generates a secret key and encrypt it with the server's public key and sends it to the server
- The server with its private key will decrypt and read the secret key
- All future communication will be done using the symmetric encryption/decryption using the secret key (faster).

How to send a Request?

1. From a Browser, enter URL (only GET)
2. From an HTML page, create form and submit button (only GET, POST)
3. Programmatically, if JavaScript, use Fetch API
4. From a software (Postman)
5. From VS code extension (Thunder Client)



1



2

```
fetch('https://dog.ceo/api/breeds/image/random')
.then(response => {
  console.log(response.headers.get('Content-Type'))
  console.log(response.headers.get('Date'))
})
▶ Promise {<pending>}
application/json
```

3

Uniform Resource Locator (URL)

- **Anchor**: jumps to a given section of a web page
- `http://www.miu.edu/download/index.html#downloads`

- **Port**: for web servers on ports other than the default 80
- `http://www.cs.miu.edu:8080/mscs/wap.txt`

- **Query string**: a set of parameters passed to a web program
- `http://www.google.com/search?q=miu&start=10`

A typical URL would have a protocol, a hostname, and a file name (path, or params).

Hypertext Markup Language(HTML)

Describes the content and structure of information on a web page with semantic **markups**:

- open/close markups
- self-closed markups

Each markup or tag, can be configured with **attributes**

Whitespace between the tags is ignored

There are many versions of HTML, the current version is HTML5

VS Code extensions

- Auto Close Tag
- Auto Rename Tag
- Live Server
- Error Lens
- Webhint
- Activate Auto-Format in VS-Code settings
- Activate Auto-Save

Learn about using the **Devtools** in Chrome.

HTML5

A standard HTML5 page has two sections:

- **`head`**: page configurations (metadata, CSS, JS, favicon, base.. etc)
- **`body`**: markups to be converted to DOM elements

Try the **`html:5`**, **`lorem`**, **`ul>li*3`** Emmets, and explore VS code Emmet Abbreviations: https://docs.emmet.io/cheat-sheet/

Block vs Inline Elements

A block element always starts on a new line and takes up the full available width (stretches out to the left and right as far as it can).

Demo: `div, span, p, h1/6, hr, br, img, a, section, header, footer, nav, aside, article, ol/li, ul/li, table/tr/th/td, strong, i, <!-- comment -->`

Tags must be correctly nested, a closing tag must match the most recently opened tag.
Don't use tables for layout, it is a semantic tag that represents an actual table of data.

Absolute vs. Relative URL

Absolute URL: A complete address containing all the parts needed to find a specific file or web page on the internet.

Relative URL: It provides directions to find a resource relative to the current webpage. A relative URL skips some parts of the absolute URL, assuming the browser can fill in the gaps based on the current webpage location.

The `<base>` tag specifies the base URL and/or target for all relative URLs in a document.

Form Elements

Forms start with **<form>** and end with **</form>**, attributes include:

`action, method, enctype, novalidate, autocomplete`

Form Elements include:

`input, textarea, button, select/option, label`

An **input** element must have a **name**, **value**, and **type**. Types include: `text, checkbox, radio, file, password, number, date, time, email, time, color.. etc`

A **button** element must have a **type**. Types include: `button, submit, reset`

Main point

The Hypertext Markup Language uses tags to demarcate different sections of a text. An HTML page always starts with a <html> tag, inside of which it has a <head> tag to describe the page and a <body> tag of the contents that will be displayed. These are the tags you will use for every HTML page. This is a foundational concept

Well begun is half done. Start with a good foundation and build upon that.

Connecting The Parts Of Knowledge With The Wholeness Of Knowledge

Layers of Abstraction

- HTML is the basis of web programming,
- To be an effective web programmer you also must understand the deeper underlying realities of HTTP, TCP, and DNS.

- **Transcendental consciousness** is when our mind is in contact with the deepest underlying reality, the unified field.
- **Impulses within the Transcendental field**: the infinite dynamism of the unified field constantly expresses itself as all of the layers of the universe
- **Wholeness moving within itself**: In Unity Consciousness, one experiences that all these layers are ultimately composed of pure consciousness, our own pure awareness.