Assignment No 1

Group B (Network Security)

Aim: Implement a client and a server on different computers using python. Perform the communication between these two entities by using RSA cryptosystem.

Objectives:

1. To learn various client/server environments to use application layer protocols.
2. To understand the network security by using public key cryptography algorithms.

Theory:

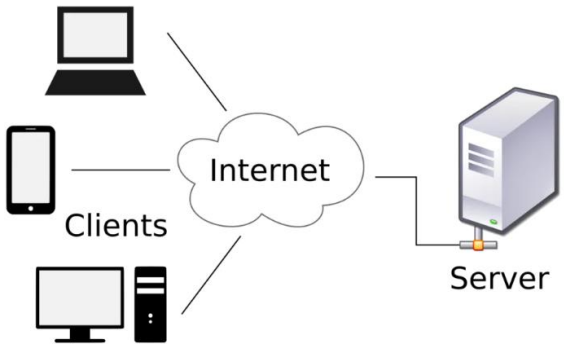
Client-Server Architecture

Client-server architecture is a network model where every process or computer on a network is a server or a client. The client-servers are the robust computers that are dedicated to managing the printers, disk drives, and network traffic. Clients are workstations or PCs on which the users run their applications. Clients mainly rely on the servers for resources, like devices, files, and processing power.

A client-server relationship corresponds to the request–response pattern and should adhere to the common communications procedure that defines the language, rules, or dialog patterns used. The client-server communication adheres to TCP or IP protocol suite.

The TCP protocol maintains the connection until a client/server has completed their message exchange. And TCP protocol decides the best method to distribute the application data in packets that networks will deliver, transfers the packets to and get packets from a network, and manages the flow control or retransmission of the dropped and garbled packets. Internet Protocol is the connectionless protocol where every packet traveling on the Internet is the independent data unrelated to other data units.

How Does Client-Server Architecture Works?



Now let us go ahead and look at how the Internet works through web browsers.

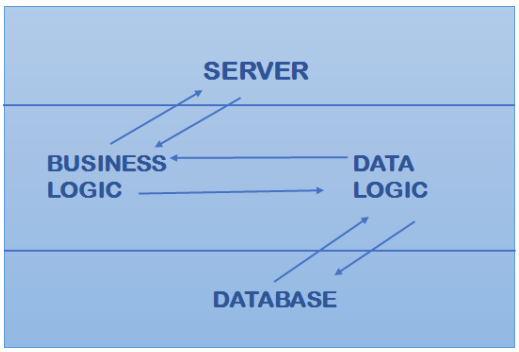
Client: The word Client means an organization or an individual using a service. Even in the digital world Client is a Host (computer) that can receive information or using service from the Servers.

Server: Server means a person that serves something. The server, in the digital world, is the remote computer that offers information or access to services.

So, it is basically a Client requesting something & a Server serving it providing its presence in a database.

Types of Client-Server Architecture

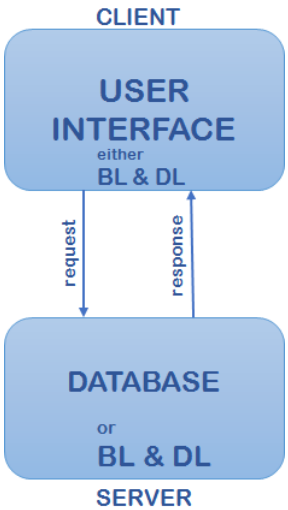
1-Tier Architecture



All client or server configuration settings, UI environment, data logic, as well as marketing logic are there on the same system. The 1-tier architecture services are quite reliable but tough tasks to handle as they have all data in various variance that will be allotted the complete replication of the whole work. 1-Tier architecture also has different layers.

For example –Business, Presentation, Data Access layer using a single software package. Data will be saved on a local machine. Some applications manage 3 tiers like an MP3 player and MS Office; however, these applications are presented in a 1-tier architecture system.

2-Tier Architecture



In 2-Tier Architecture, the UI is stored at the client machine, and the database gets stored on a server. The business logic and database logic are filed at server or client but have to be well-maintained.

Suppose Data Logic and Business Logic are collected at the client-side, it's called fat client-server architecture. Suppose Data Logic and Business Logic are handled on a server, its thin client-server architecture. It is considered affordable.

In 2-Tier architecture, server and client need to come in the direct incorporation. Suppose a client provides any input to a server there must not be any intermediate. It is generally done for rapid results and to avoid confusion between various clients. For example, an online ticket reservations application uses this 2-Tier architecture.

3-Tier Architecture

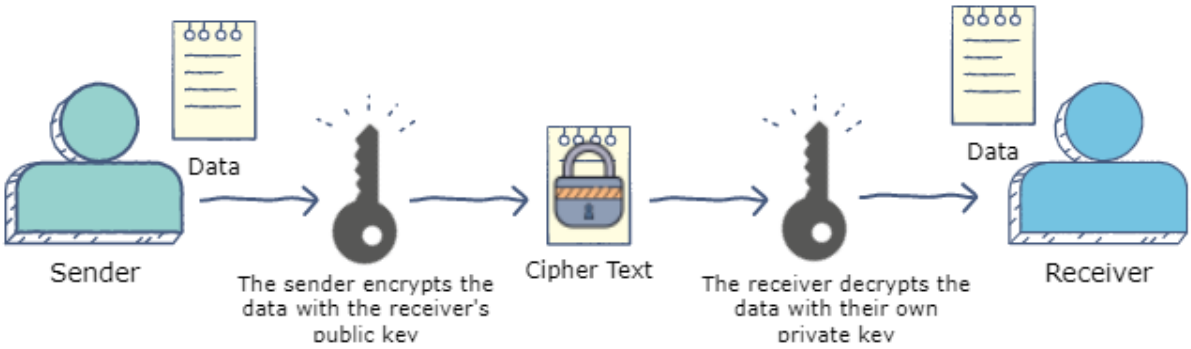
It consists of the presentation tier that is the User Interface layer, an application tier that is a service layer, which performs the detailed processing, and a data tier that consists of the database server, which stores information. Three-tier architecture can be split into 3 parts, the presentation layer (or Client Tier), the Application layer (or Business Tier), and the Database layer (or Data Tier). It works in the following ways: The Client system handles the Presentation layer; the Application server looks after the Application layer, and the Server system supervises the Database layer.

What is the RSA algorithm?

The RSA algorithm is an asymmetric cryptography algorithm; this means that it uses a public key and a private key (i.e two different, mathematically linked keys). As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone.

The RSA algorithm is named after those who invented it in 1978: Ron Rivest, Adi Shamir, and Leonard Adleman.

The following illustration highlights how asymmetric cryptography works:



How it works

The RSA algorithm ensures that the keys, in the above illustration, are as secure as possible. The following steps highlight how it works:

1. Generating the keys

1. Select two large prime numbers, xx and yy. The prime numbers need to be large so that they will be difficult for someone to figure out.
2. Calculate n = x$n=x$ x yy.
3. Calculate the ***totient*** function; $\phi(n)=(x-1)(y-1)$.
4. Select an integer ee, such that ee is ***co-prime*** to $\phi(n)$ and

   $1 < e < \phi(n)$. The pair of numbers (n,e) makes up the public key.

5. Calculate d such that e.d = 1 mod $\phi(n)$.

d can be found using the extended euclidean algorithm. The pair (n,d) makes up the private key.

2. Encryption

Given a plaintext PP, represented as a number, the ciphertext C is calculated as:

$$C = P^e \bmod n.$$

3. Decryption

Using the private key (n,d)(n,d), the plaintext can be found using:

$$P = C^d \bmod n.$$

Pseudocode

Consider an example of the RSA algorithm through the following pseudocode:

```
int x = 61, int y = 53;
int n = x * y;
// n = 3233.

// compute the totient, phi
int phi = (x-1)*(y-1);
// phi = 3120.
```

int e = findCoprime(phi);

// find an 'e' which is > 1 and is a co-prime of phi.

// e = 17 satisfies the current values.

// Using the extended euclidean algorithm, find 'd' which satisfies

// this equation:

d = (1 mod (phi))/e;

// d = 2753 for the example values.

public_key = (e=17, n=3233);

private_key = (d=2753, n=3233);

// Given the plaintext P=123, the ciphertext C is :

C = (123^17) % 3233 = 855;

// To decrypt the cypher text C:

P = (855^2753) % 3233 = 123;