Mohamed Khider University of Biskra
Faculty of Sciences and Technology
Department of Electrical engineering

MASTER THESIS

Electrical Engineering
Telecommunications
Networks and Telecommunication

Submitted and Defended by:
Mr. SAYAH Tarek

On : Tuesday, 26 June 2018

Conception of Client/Server System with Encrypted Data Exchanging Using Sockets

Academic Year : 2017 - 2018

الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي و البحث العلمي
Ministry of higher education and scientific research

University Mohamed Khider Biskra
Faculté of Science and Technology
Department of Electrical Engineering
Section : Electronic
Option : Networks and Communications

Thesis Presented to obtain
the Academic Diploma in

MASTER

Theme

Conception of Client/Server System with Encrypted Data Exchanging Using Sockets

Presented By:
SAYAH Tarek

Favorable opinions of Supervisor:

Favorable opinions of President of jury:

Stamp and signature

الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي و البحث العلمي
Ministry of higher education and scientific research

University Mohamed Khider Biskra
Faculté of Science and Technology
Department of Electrical Engineering
Section : Electronic
Option : Networks and Communications

Theme

Conception of Client/Server System with Encrypted Data Exchanging Using Sockets

Proposed by : SAYAH Tarek
Directed by : GUESBAYA Tahar

Abstract

Information security and computer networks allow keeping the confidentiality of data in order to facilitation of access to it with more privacy.

Among the most models that rely on networks in sending data the client/server model, for ease for transmission and it is the model that is more compatible with most operating systems and methods of cryptography.

From the theoretical side, we will present the most important titles that explain the concept of client/server and its components, and giving an overview of classical and modern encryption methods.

In addition to designing a client/server model where we can encrypt files sent, and we will apply the advanced encryption standard, which is one of the latest encryption methods provided by the cryptography.

تلخيص

أمان المعلومات وشبكات الكمبيوتر هو مجال يسمح للبيانات بالاحتفاظ بسريتها للتحكم في الوصول إليها بسهولة وخصوصية أكبر ;و من بين أهم النماذج التي تعتمد عليها الشبكات في إرسال البيانات هو نموذج العميل/ الخادم إذ يوفر سهولة في الإرسال ويعتبر من أكثر النماذج توافقا مع أنظمة التشغيل وأغلب أساليب الحماية.

من الجانب النظري سنقدم أهم العناوين التي توضح لنا مفهوم العميل/الخادم ومكوناته مع إعطاء نظرة شاملة عن أساليب التشفير قديما و حديثا.

في الجانب التطبيقي سنقوم بتصميم نموذج العميل /الخادم الذي يمكننا من تشفير الملفات ثم إرسالها، و سنقوم بتطبيق معيار التشفير المتقدم الذي يعتبر من أحدث أساليب الحماية التي يوفرها علم التشفير .

Dedication

I would like to dedicate this project to many people that were beside me, without their unrelenting confidence and support I couldn't have come so far.

*To my **mother** and my **father**, thank you for your endless love, support, encouragement, and for teaching me how to live intelligently and how to be honest with myself and with the others, thank you for your patience, understanding, I love you so much, and I will always remember your words especially when I lose the control, you are the best parents in the world, I'm forever grateful to you, may God protect you.*

*Also, I would like to mention all my friends **Mohamed El Amine**, **Walid**, **Nadji**, **Mohamed**,**Riyad** without you guys this project would not have been possible, you are so special to me, and I will remember every thing about you, especially our laughs, and our jocks, for me you are my best friends and my another family, I really wish for all of you to make your dreams come true and to get the best of the best in your lives, I love you so much guys, may God protect you.*

Tarek SAYAH

Acknowledgements

I would like to thank my teacher and supervisor **Dr. Guesbaya Taher** *for offering this interesting subject for my graduation memorandum. Despite of his limited time, Dr. Guesbaya has been always ready to provide advice, guidance and help during this entire year.*

My gratitude and respect also go out to all my teachers at the University of Mohamed Khider Biskra. Thanks to them and their great lectures.

Last but not least, many thanks to all my family members and my friends who were always there to support me.

Tarek SAYAH

Table of Contents

Chapter 3: Cryptography

List of Figures

List of Tables

List of Abbreviations

OSI	Open Systems Interconnection
PC	Personal Computer
DBMS	DataBase Management System
CPU	Central Processing Unit
ISO	International Organization for Standardisation
MAC	Media Access Control
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
API	Application Programming Interface
IPC	Inter Process Communications
BSD	Berkeley Software Distribution
FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
OS	Operating System
IPV4	Internet Protocol version 4
IPV6	Internet Protocol version 6
DNS	Domain Name System
IMAP	Internet Mail Access Protocol

IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
DES	Data Encryption Standard
NIST	National Institute of Standards and Technology
FIPS	Federal Information Processing Standard
ECM	Electronic Codebook Mode
CBC	Cipher Block Chaining Mode
OFM	Output Feedback Mode
CFM	Cipher Feedback Mode
AES	Advanced Encryption Standard
NSA	National Security Agency
IDE	Integrated Development Environment

General Introduction

Many companies now prepare for the day they will be breached rather than expect technology to keep them safe and secure all the time. Often attackers can get into a corporate network using stolen staff credentials but that just gets them a foothold. From there they need to explore, expand and gather network privileges that help them get at the data they really want to steal, Also accessing personal data of ordinary people in different ways. We do not care how they access to this as much as we are concerned about the status of this data in terms of protection.

The most reliable way to ensure the safety of a company's computers is to refrain from putting them on a network and to keep them behind locked doors. Unfortunately, however, that is not a very practical solution, and here companies and people need to encrypt data because if the hackers can steal that data they will not able to open or use it.

Today, computers are most useful if they are connected together to share information and resources, and companies that put their computers on a network need to take some simple precautions to reduce the risk of unauthorized access. Client/server model and cryptography are very homogeneous domains and provides useful services that enables us to safety use of networks.

This project has an objective to conceive and implement an efficient application that use cryptography as an instrument to encrypt data exchanging in a client/server system and we will choose ever we need to achieve our goal .

Our study is composed of four chapters:

In the first chapter, we will present the client/server model in general, and then we will discuss its components and the client/server advantages and disadvantages with mention of its topologies and types. Finally, we will describe the OSI and TCP/IP models and will show how the client/server needs the APIs.

In the second chapter, we will present a general introduction to sockets, its importance and benefits, also we will give some definitions about the sockets and its perimeter which include the sockets types and connection modes and how the sockets using addresses. However, we will focus principally on primary procedures that sockets used to execute.

In the third chapter, we will present the general introduction about cryptography, its classical and modern methods, and how the cryptography save our privacy over the times.

In the fourth chapter, we will show the general conception of our application, then we will look at the detailed description of each approach have been used. Then we will log in the implementation section, which include tools and the language programming that we will use, also we will show the application details.

Chapter 1

Client/Server Model

1.1 Introduction

The term Client/Server was first used in the 1980s in reference to personal computers on a network. The actual Client/Server model started gaining acceptance in the late of 1980s.The term Client/Server is used to describe a computing model for the development of computerized systems.

This model is used on the distribution of functions between two types of independent and autonomous entities: Server and Client.

A Client is any process that request specific services from server processes. A Server is process that provides requested services for Clients. Client and Server processes can reside in same computer or in different computers linked by a network.

In this chapter, we first present the client/server paradigm in general. Then we discuss the various characteristics, topologies, Components of the client and the server. We then describe the OSI and TCP/IP Models .Finaly we conclude the chapter with some needs of the client/server model.

1.2 Definition of Client/Server Model

The client/server model is built around a network which include two connected computers, the server and the client, [1]Typically, a client sends a request to a server, and the server returns a response to the client. [2]

The client sends during the data processing one or more requests to the servers to perform specified tasks. The server part provide services for the clients. [3]

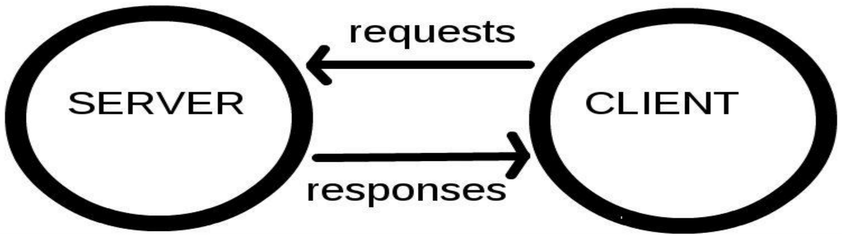


Figure 1.1: A Simplified Diagram of How the Client/Server Interacts.

1.3 Characteristics of The Client/Server

Although minor variations exist, most instances of client/server interaction have the same general characteristics. In general, client software:

- Is an arbitrary application program that becomes a client temporarily when remote access is needed, but also performs other computation.
- Is invoked directly by a user, and executes only for one session
- Runs locally on a user’s personal computer
- Actively initiates contact with a server
- Can access multiple services as needed, but usually contacts one remote server at a time
- Does not require especially powerful computer hardware

In contrast, server software:

- Is a special-purpose, privileged program dedicated to providing one service that can handle multiple remote clients at the same time
- Is invoked automatically when a system boots, and continues to execute through many sessions
- Runs on a large powerful computer (powerful hardware and a sophisticated operating system)
- Waits passively for contact from arbitrary remote clients
- Accepts contact from arbitrary clients, but offers a single service
- Requires powerful hardware and a sophisticated operating system [4]

1.4 Requests, Responses, and Direction of Data Flow

The terms client and server arise from which side initiates contact. Once contact has been established, two-way communication is possible. In some cases, a client sends a series of requests and the server issues a series of responses. The concept can be summarized:
Information can flow in either or both directions between a client and server. Although many services arrange for the client to send one or more requests and the server to return responses. [4]

1.5 The Different Client/Server Models

In fact, the differences are essentially related to the services that are provided by the server. we currently distinguish:

1.5.1 The Client/Server of Data

In this case, the server ensures management tasks, storage and data processing. This is the best-known case of client/server to use by all major DBMS.

The database with all its tools (maintenance, backup ....) is installed on a server workstation. On the clients, an access software is installed allowing access to the database of the server; all data processing is performed on the server, which returns the information requested by the client.

1.5.2 The Client/Server of Presentation:

In this case, the presentation of the pages displayed by the client is fully supported by the server. This organization presents the disadvantage of generating a strong network traffic.

1.5.3 The Client/Server of Processing:

In this case, the server performs processing at the request of the client. It may be special data processing, input form verification, alarm processing.

These processes can be realized by programs installed on servers but also integrated into databases, in this case, the given and treatment part are integrated [5]

1.6 Client/Server Topologies

A Client/Server topology refers to the physical layout of the Client/Server network in which all the clients and servers are connected to each other. The possible Client/Server topological design and strategies used are as follows:

1. Single client, single server.
2. Multiple clients, single server.
3. Multiple clients, multiple servers. [6]

1.6.1 Single Client, Single Server:

This topology is shown in the Fig. 1.2 given below. In this topology, one client is directly connected to one server.

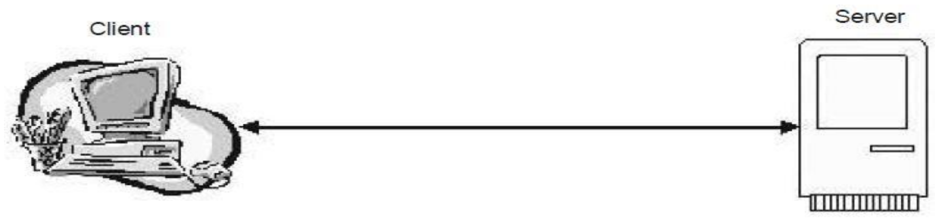


Figure 1.2: Single Client, Single Server. **[6]**

1.6.2 Multiple Clients, Single Server:

This topology is shown in the Fig. 1.3 given below. In this topology, several clients are directly connected to only one server.

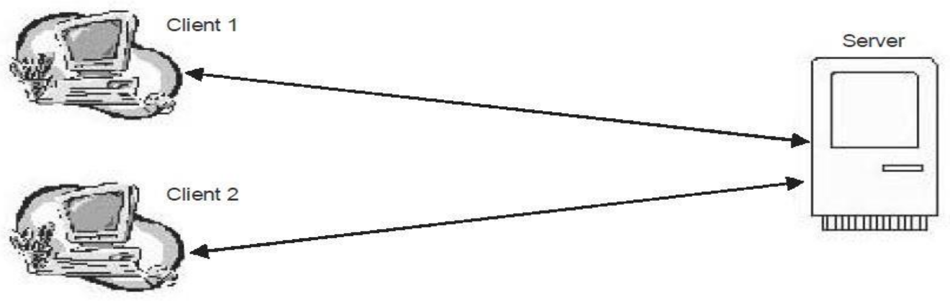


Figure 1.3: Multiple Clients, Single Server. **[6]**

1.6.3 Multiple Clients, Multiple Servers:

This topology is shown in the following Fig. 1.4 In this topology several clients are connected to several servers.

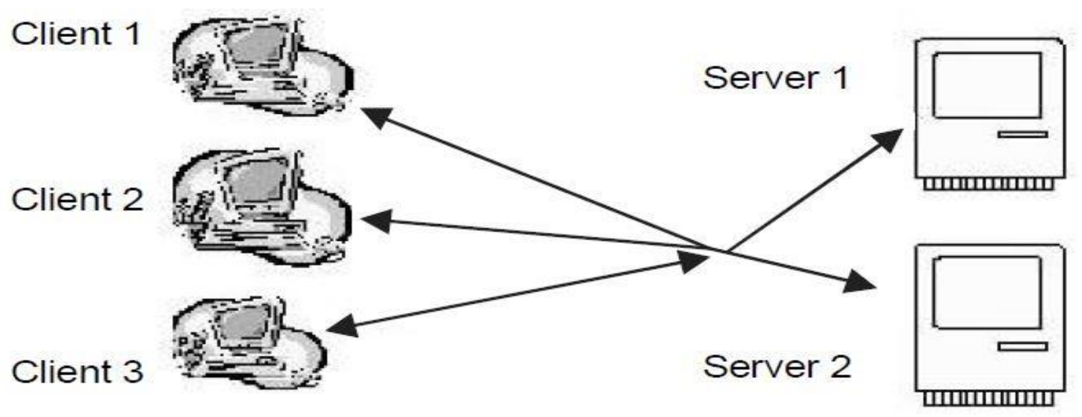


Figure 1.4: Multiple Clients, Multiple Servers. **[6]**

1.7 Types of Client/Server Architecture:

Broadly, there are three types of Client/Server systems in existence:

1.7.1 Two-tiers Client/Server Model

In this type of architecture, the workload is divided between the server (host of the system) and the client (which hosts the user interface).

In reality, these are located on separate computers but there is no absolute requirement of this, providing that the tiers are logically separated can be hosted on the same computer.

1.7.1.1 Advantages

- Ease in Developing Applications: Applications can easily be developed due to simplicity.
- User Satisfaction: Maximum user satisfaction is gained with accurate and fast prototyping of applications through robust tools.
- Applicable for Homogeneous Environment: Since this contains static business rules, it's more applicable for homogeneous environment.
- High Performance: Database server and business logic is physically close, which offers higher performance.

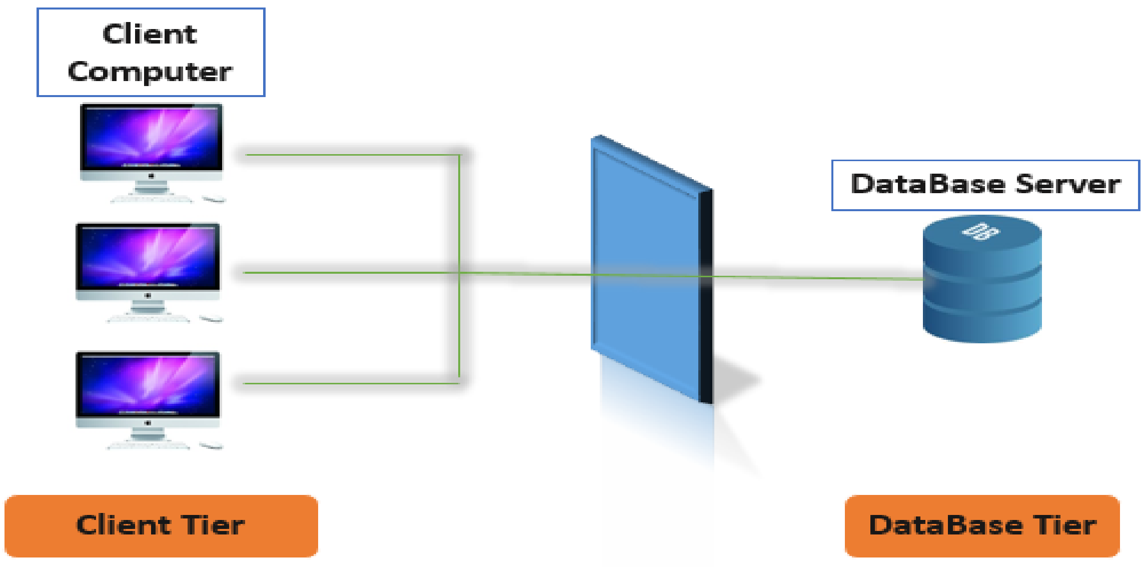


Figure 1.5: Two-tier Client/Server Model

1.7.2 Three-tiers Client/Server Model

To overcome the limitations of the two-tier architecture, an additional tier was introduced. The purpose of the additional tier (called as "middle" or "rule" tier) is handle application execution and database management, as with the two-tier model, the tiers can either be implemented on different physical machine, or multiple may be co-hosted on a single machine.

In the three-tier architecture the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platform.

The three-tiers in a three-tier architecture are:

- Presentation Tier: Occupies the top level and communicates with other tiers by sending results to the browser and other tiers in the network.
- Application Tier: Also called the middle tier, logic tier or business logic, this tier is pulled from the presentation tier. It controls application functionality by performing detailed processing.
- Data Tier: Houses database server where information is stored and retrieved. Data in this tier is kept independent of application servers or business logic.

1.7.2.1 Advantages

- Improved Data Integrity: Data corruption through client application can be eliminated as the data passes through the middle tier for updating database ensures its validity.
- Enhanced Security: The placement of business logic on a centralized server makes the data more secure.
- Hidden Database Structure: The actual structure of database often remains hidden from clients enabling any change in the database to be hidden.

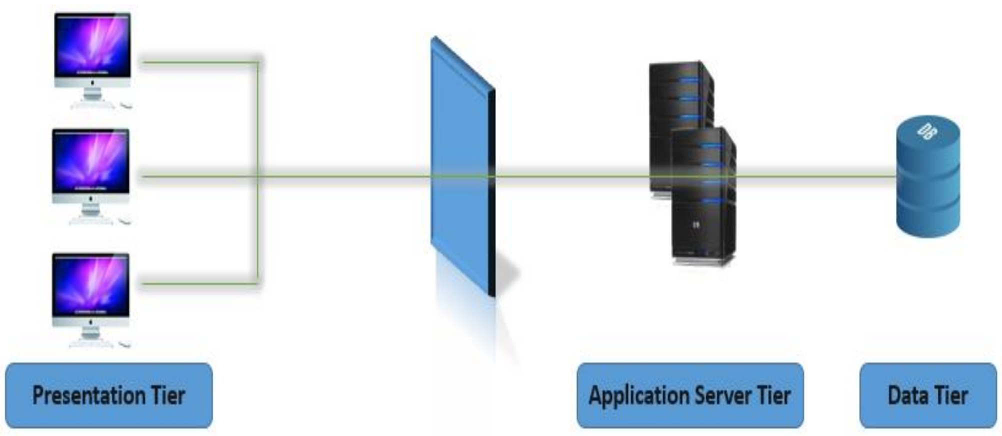


Figure 1.6: Three-tier Client/Server Model.

1.7.3 N-tiers Client/Server Model

N-tier architecture obliges developer to design components according to a business schema that represents entities, relationship, activities roles, and rules, thereby enabling them to distribute functionality across logical and physical tiers, allowing better utilization of hardware and platform resources, as well as sharing of those resources and the components that they support to serve several large applications at the same time. [6]

1.7.3.1 Advantages

- Overall performance has been improved.
- The business logic is centralized.
- Enhanced security level is attained.

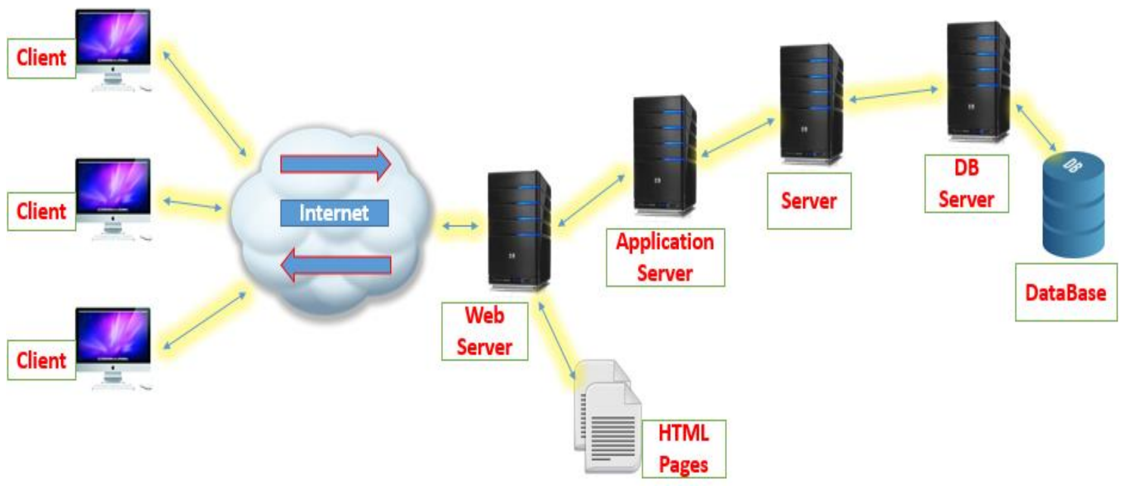


Figure 1.7: N-tier Client/Server Model.

1.8 Client/Server Adventages And Desadvantages

1.8.1 Advantages

There are various advantages associated with Client/Server computing model.

- Performance and reduced workload: Processing is distributed among the client and server unlike the traditional PC database. Some database servers can even store and run procedures and queries on the server itself, reducing the traffic even more.
- Workstation independence: Users are not limited to one type of system or platform.

- System interoperability: Client/Server computing not only allows one component to be changed, it also makes it is possible for different type of components systems to work together.
- Scalability: The modular nature of the Client/Server system may be replaced without adversely affecting the rest of the system.
- Data accessibility (enhanced data sharing): Since the server component holds most of data in a centralized location, multiple users can access and work on the data simultaneously.
- System administration (centralized management): Client/Server environment is very manageable.
- Location independence of data processing: Users log into an application from the desktop with no concern for the location or technology of the processors involved.

The client/server model was originally developed to allow more users to share access to database applications. Compared to the mainframe approach, client/server offers improved scalability because connections can be made as needed rather than being fixed.

The client/server model also supports modular applications that can make the job of creating software easier. In so-called "two-tier" and "three-tier" types of client/server systems, software applications are separated into modular pieces, and each piece is installed on clients or servers specialized for that subsystem.

Client/server networks generally offer advantages in keeping data secure. [7] And it requires less computational power on the client side. However, this has been somewhat circumvented due to ever-increasing CPU power and therefore most desktop PCs are ludicrously overpowered to operate as simple clients, e.g., for browsing and email. [8]

1.8.2 Disadvantages

There are various disadvantages associated with the Client/Server computing model.

- Maintenance cost: Major disadvantages of Client/Server computing is the increased cost of administrative and support personnel to maintain the database server.
- Training cost: Training can also add to the start-up costs as the DBMS may run on an operating system that the support personnel are unfamiliar with.

- Hardware And Software cost: There is also an increase in hardware costs. And overall cost of the software that usually higher than that of traditional PC based multi-user DBMS. [6]

1.9 Client/Server Components

1.9.1 Components

Client/Server architecture is based on hardware and software components that interact to form a system. The system includes mainly three components:

1- Hardware (client and server).

2- Software (which make hardware operational).

3- Communication middleware.

1.9.2 Interaction Between the Components

The interaction mechanism between the components of Client/Server architecture is clear from the Figure 1.10, the client process is providing the interface to the end users. Communication middleware is providing all the possible support for the communication taking place between the client and server processes. Communication middleware ensures that the messages between clients and servers are properly routed and delivered. Requests are handled by the database server, which checks the validity of the request, executes them, and send the result back to the clients. [6]

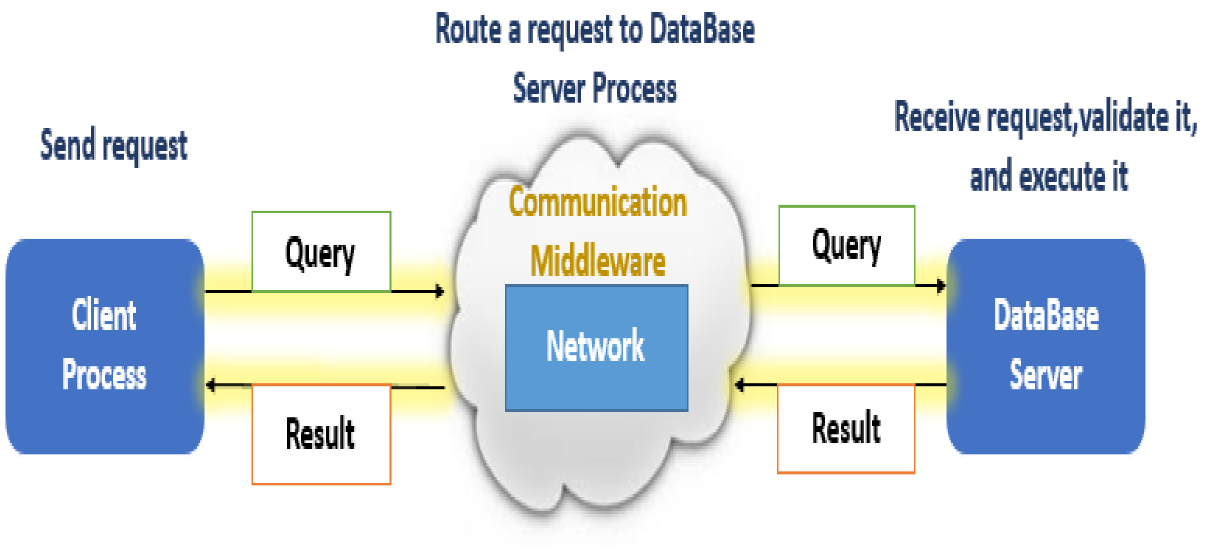


Figure 1.8: Components Interaction.

1.10 OSI and TCP/IP Models

1.10.1 OSI Model

The OSI model was developed by the International Organization for Standardization (ISO) in 1984. It is now considered the primary Architectural model for inter-computer communications.
The OSI model is a logical model of communications for networked devices and software. The OSI model does not specify the protocols that should be used at each of the seven layers, but it specifies the functionality that should be provided by those protocols. [9]

The OSI Reference Model is composed of seven layers, each specifying particular network functions. Each layer gives a service to the layer above it in the protocol specification. And Each layer communicates with the same layer's software or hardware on other computers.

The seven layers provides the following services:

- **Layer 7: Application**

The application layer establishes the availability of intended communication partners, synchronizes and establishes agreement on procedures for error recovery and control of data integrity.

- **Layer 6: Presentation**

The presentation layer translates between multiple data formats by using a common format. And provides encryption and compression of data.

- **Layer 5: Session**

The session layer defines how to start, control and end conversations (called sessions) between applications. It also synchronizes dialogue between two hosts' presentation layers and manages their data exchange.

- **Layer 4: Transport**

The transport layer regulates information flow to ensure end-to-end connectivity between host applications reliably and accurately. And segments data from the sending host's system and reassembles the data into a data stream on the receiving host's system.

- **Layer 3: Network**

Defines logical addressing so that any endpoint can be identified. And defines how routing works and how routes are learned so that the packets can be delivered. It also gives how to fragment a packet into smaller packets to accommodate different media.

- **Layer 2: Data Link**

The data link layer provides access to the networking media and physical transmission across the media and this enables the data to locate its intended destination on a network. And provides reliable transit of data across a physical link by using the Media Access Control (MAC) addresses.

- **Layer 1: Physical**

The physical layer deals with the physical characteristics of the transmission medium. It defines the electrical, mechanical, procedural, and functional specifications for activating, maintaining, and deactivating the physical link between end systems. [9]

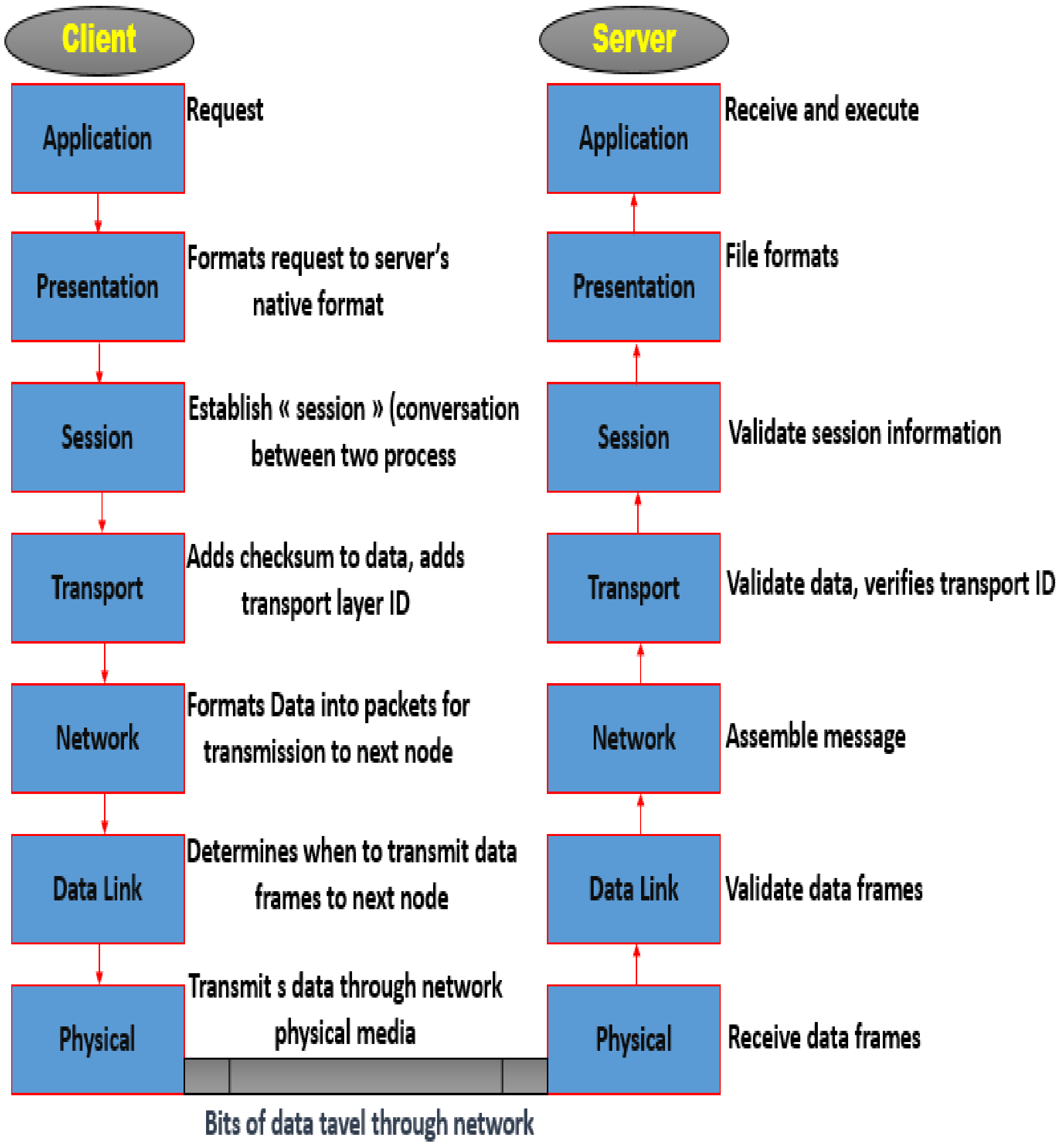

Client
Server
Application
Request
Application
Receive and execute
Presentation
Formats request to server's native format
Presentation
File formats
Session
Establish « session » (conversation between two process
Session
Validate session information
Transport
Adds checksum to data, adds transport layer ID
Transport
Validate data, verifies transport ID
Network
Formats Data into packets for transmission to next node
Network
Assemble message
Data Link
Determines when to transmit data frames to next node
Data Link
Validate data frames
Physical
Transmit s data through network physical media
Physical
Receive data frames
Bits of data tavel through network

Figure 1.9: Flow of Information Through the OSI Model.

1.10.2 TCP/IP Model

Unlike the OSI model's seven layers, the TCP/IP model contains only four layers.

The four layers of the TCP/IP model are:

- ■ Application layer
- ■ Transport layer
- ■ Internet layer
- ■ Link layer

The TCP/IP Application layer can be said to encompass the OSI model's Application and Presentation layers. In other words, Layer 4 of the TCP/IP model performs the services required of Layer 6 and Layer 7 of the OSI model. [10]

The TCP/IP Transport layer provides host-to-host or end-to-end communications.Transport layer protocols may provide reliable or nonguaranteed data delivery. TCP is an example of a Transport layer protocol that provides reliable delivery of data (usually called segments), and UDP is an example of a protocol that provides nonguaranteed delivery of data (usually called datagrams). [8]

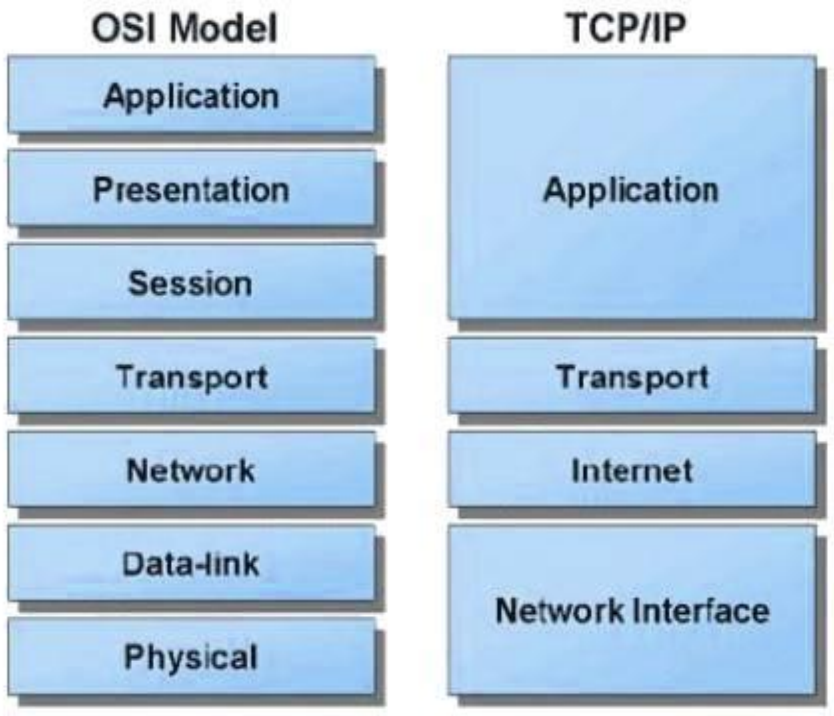


Figure 1.10: OSI and TCP/IP Models.

1.11 Middleware

1.11.1 Introduction to middleware

Any time we split an application into components and place those components on separate computers, we must provide a way for them to communicate. Middleware is a general term for all the software components that allow us to connect separate layers or components into complete distributed systems. As such, it is often referred to as the slash (/) in client/server. Since the slash can have a different meaning depending on the distribution model, it turns out (not surprisingly) that we need different kinds of middleware to implement each of the different physical models we have been discussing. [11]

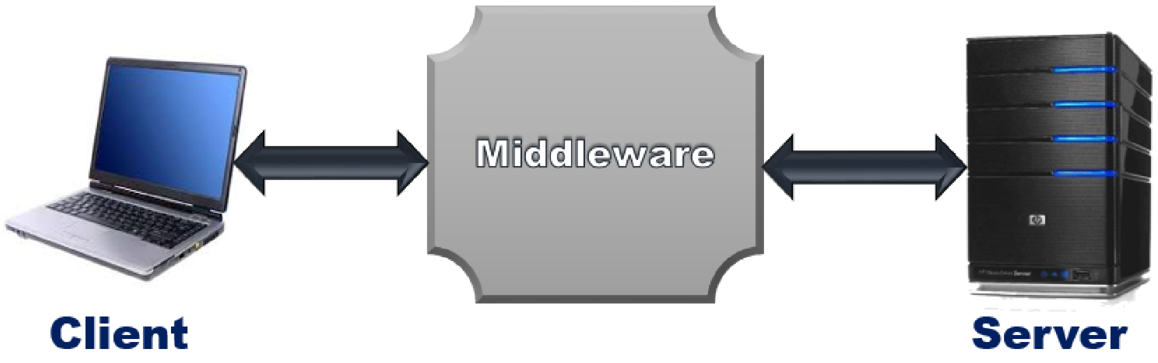


Figure 1.11: Simple Architecture of Middleware.

1.11.2 Middleware Services

A middleware likely to make the following services:

- Conversion: Services used for machine-to-machine communication using different data formats.
- Addressing: Identifies the server machine on which the requested service is located to derive the access path. As far as possible.
- Security: Ensures confidentiality and security of data by means of authentication and encryption mechanisms.
- Communication: Allows the transmission of messages between the two systems without alteration. [5]

The middleware masks the complexity of inter-application exchanges and thus makes it possible to raise the level of the APIs used by the programs. Without this mechanism, the programming of a client/server application would be complex and difficult to evolve.

1.12 APIs

1.12.1 Definition

An application programming interface (API) is a specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables. An API specification can take many forms, including an International Standard such as the libraries of a programming language, e.g. Standard Template Library in C++ or Java API [12]. In other words, it is a set of functions or procedures used by computer programs to access operating system services, software libraries, or other systems.

1.12.2 API in object-oriented languages

In object-oriented languages, an API usually includes a description of a set of class definitions, with a set of behaviors associated with those classes. This abstract concept is associated with the real functionality exposed, or made available, by the classes that are implemented in terms of class methods. [12]

1.12.3 Multiple Service Interaction

APIs needed a common interface for consumption and interaction between different services. Traditional API interfaces, be it text documentation, or others like Javadoc's, do not allow for interactions between services in different languages.
For example, socket services defined with client/server can communicate with each other irrespective of the language they are built in, since it is language agnostic and machine-readable. [13]
Programming Interface Mechanism Allows programs to exchange data. Client / server applications only see communication layers through socket APIs [5]

1.12.4 Language used

An API can be:

- language-dependent: meaning it is only available by using the syntax and elements of a particular language, which makes the API more convenient to use.
- language-independent: written so that it can be called from several programming languages. This is a desirable feature for a service-oriented API that is not bound to a specific process or system and may be provided as remote procedure calls or web services. [12]

1.12.5 Goals of APIs

The broad goals of API design in general can be defined as:

- Enabling self-service for app developers and app users alike
- Reducing barriers to accessing valuable enterprise resources
- Prioritizing the needs and preferences of client app developers
- Encouraging collaboration between and among internal and external resources
- Addressing the security and scaling issues of exposing IT assets to the open market [8]

1.12.6 Socket API

The socket API is an Inter-Processing Communication (IPC) programming interface provides a programming construct termed a socket.

A process wishing to communicate with another process must create an instance, or instantiate, such a construct, the two processes then issues operations provided by the API to send and receive data. [14]

1.12.6.1 Connection-oriented & connectionless datagram socket APIs

A socket programming construct can make use of either the UDP or TCP protocol.

- Sockets that use TCP are termed *stream sockets*.
- Sockets that use UDP for transport are known as *datagram sockets*

➢ **Connection-oriented Socket API (*Stream-mode Socket*)**

- A stream-mode socket is established for data exchange between two specific processes.
- Data stream is written to the socket at one end, and read from the other end.
- A stream socket cannot be used to communicate with more than one process.

➢ **Connectionless Socket API (*datagram-mode Socket)***

- Possible for multiple processes to simultaneously send datagrams to the same socket established by a receiving process

  The order of the arrival of these messages will be unpredictable, in accordance with the UDP protocol [15].

1.13 Conclusion

New challenges and developments afford new opportunities, Computer networks have many uses, both for companies and for individuals, in the home and while on the move. Companies use networks of computers to share corporate information, typically using the client/server model with employee desktops acting as clients accessing powerful servers in the machine room. For individuals, networks offer access to a variety of information and entertainment resources, as well as a way to buy and sell products and services.

In this chapter, we briefly presented the client/server model as an instrument to facilitate communication, as well as a detailed explanation of its types and models, Finaly we concluded the chapter with an introduction to sockets and APIs.

In the next chapter, we will focus on the sockets as an interprocess communication and its various types. In addition, this chapter will reviews the global explanation of sockets

Chapter 2

Sockets

2.1 Introduction

Network is probably one of the features that is most used in the current world. As soon as people want to send or receive data over a network in a program, Client/server communications needs to an API to connect and do responses on each side, APIs offering an abstract that we call it sockets.

In this chapter, we take an in-depth look at the sockets programming facility, including standard UDP and TCP client/server programming, fine tuning sockets with socket options and touch upon the use of sockets. Then we will mention some details about sockets types and its diferente modes.

2.2 A Brief Historical Introduction

The Berkeley Software Distribution (BSD) sockets programming facility is a collection of programming language functions and data types, which is known as the BSD sockets application programming interface (API). This facility was first introduced with the BSD UNIX operating system in the early 1980s, but is now available on most UNIX-like operating systems and supported on the Microsoft Windows platform. [16]

2.3 Understanding Sockets

The Socket API provides a programming construct called a “socket”. A process wishing to communicate with another process must create an instance or instantiate a socket. Two processes wishing to communicate can instantiate sockets and then issue operations provided by the API to send and receive data. Note that in network parlance, a packet is the unit of data transmitted over the network. Each packet contains the data (payload) and some control information (header) that includes the destination address.

A socket is uniquely identified by the IP address of the machine and the port number at which the socket is opened (i.e. bound to). Port numbers are allocated 16 bits in the packet headers and thus can be at most 66535. Well-known processes like FTP, HTTP and etc., have their sockets opened on dedicated port numbers (less than or equal to 1024). Hence, sockets corresponding to user-defined processes have to be run on port numbers greater than 1024. [17]

2.4 Defining a Socket

Sockets are an Application Programming Interface (API) for Inter-Process-Communication (IPC). [18]This interface is most commonly used by programmers to implement network based communication between one or more computers. [16]

Socket is an end-point of communication, which identifies a local “process” at one end of a communication association. communication association is identified by two half associations {protocol, local-address, local-port, remote-address, remote-port} [19]

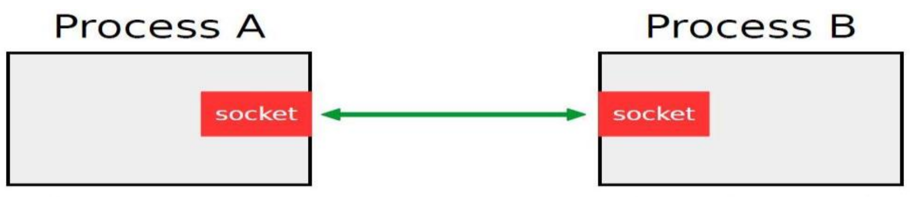


FIGURE 2.1: Socket End on Each Process.

2.5 Using Sockets

The sockets API is widely used in conjunction with the JAVA programming language to implement TCP or UDP support in software. Two basic types of applications use the sockets API: client and server. Client applications use the API to create an endpoint for communication and to initiate communication with remote server applications. Server applications, in turn, waiting for communication from remote client applications. [16]

2.6 Interprocess communication (IPC)

Interprocess communication IPC is the backbone of distributed computing. Processes are runtime representations of a program. IPC refers to the ability for separate, independent processes to communicate among themselves to collaborate on a task. Figure 2.2 illustrates basic IPC.

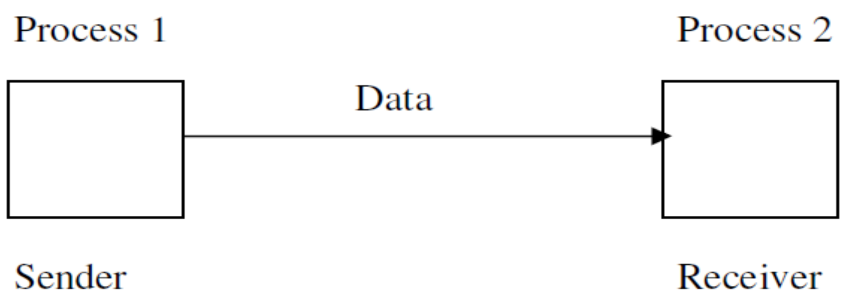


FIGURE 2.2: Simple Design of an Interprocess Communication.

Most of the operating systems (OS) like UNIX and Windows provide facilities for IPC. The system-level IPC facilities include message queues, semaphores and shared memory. It is possible to directly develop network software using these system-level facilities.

The Socket API is a low-level programming facility for implementing IPC. The upper-layer facilities are built on top of the operations provided by the Socket API. The Socket API was originally provided as part of the Berkeley UNIX OS, but has been later ported to all operating systems including Sun Solaris and Windows systems. [17]

2.7 Using Addresses

In TCP/IP socket, it takes two pieces of information to identify a particular program: An Internet address, used by IP, and a port number, the additional address interpreted by the transport protocol (TCP or UDP).

Internet addresses are binary numbers. They come in two flavors, corresponding to the two versions of the Internet Protocol that have been standardized. The most common is version 4 (IPv4); the other is version 6 (IPv6), which is just beginning to be deployed. [20]

2.7.1 IP Addresses

Different conventions are used for the two versions of IP. IPv4 addresses are conventionally written as a group of four decimal numbers separated by periods (e.g., 10.1.2.3); this is called the dotted-quad notation. The four numbers in a dotted-quad string represent the contents of the four bytes of the Internet address—thus, each is a number between 0 and 255.

The 16 bytes of an IPv6 address, on the other hand, by convention are represented as groups of hexadecimal digits, separated by colons (e.g., 2000:fdb8:0000:0000:0001:00ab:853c:39a1). Each group of digits represents 2 bytes of the address; leading zeros may be omitted, so the fifth and sixth groups in the foregoing example might be rendered as just :1:ab:. Also, one sequence of groups that contains only zeros may be omitted altogether. So the example above could be written as 2000:fdb8::1:00ab:853c:39a1.

2.7.2 Port Numbers

Both the TCP and UDP protocols use 16 bits identifiers called ports to uniquely identify the processes involved in a socket. These numbers are managed by the Internet Assigned Numbers Authority (IANA). [21]

Port numbers between 1 and 1023 are reserved for well-known services like FTP, HTTP, DNS, and IMAP. On Unix systems, including Linux and Mac OS X, only programs running as root can receive data from these ports, but all programs may send data to them. On Windows, any program may use these ports without special privileges. [22]

Ports 1024 – 65535 are available for use by programs, but other network applications maybe running and using these port numbers as well, so we not make assumptions about the availability of specific port numbers. [23]

2.7.3 Specifying Addresses

Applications using sockets need to be able to identify the remote endpoint(s) with which they will communicate, the sockets layer sometimes needs to pass addresses to the application. For example, a feature analogous to "Caller ID" in the telephone network lets a server know the address and port number of each client that communicates with it.

2.8 Primary Sockets -Procedures

- socket(): create a new socket and return its descriptor.
- bind(): associate a socket with a port and address.
- listen(): establish queue for connection requests.
- accept(): accept a connection request.
- connect(): initiate a connection to a remote host.
- recv(): receive data from a socket descriptor.
- send(): send data to a socket descriptor.
- close(): close of a socket descriptor.

2.9 Socket Types

Two types of transport service via socket API:

- reliable, byte stream-oriented
- unreliable datagram

2.9.1 The Connection-Oriented Socket (TCP)

A TCP connection is an abstract two-way channel whose ends are each identified by an IP address and port number. Before being used for communication [20], a TCP connection must go through a setup phase, which starts with the client's TCP sending a connection request to the TCP server.

TCP is designed to detect and recover from the losses, duplications, and other errors that may occur in the host-to-host channel provided by IP. TCP provides a reliable byte-stream channel, so that applications do not have to deal with these problems.

It is a connection-oriented protocol, before using it to communicate, two programs must first establish a TCP connection, which involves completing an exchange of handshake messages between the TCP implementations on the two communicating computers.

Using TCP is also similar in many ways to file input/output (I/O) [20]. In fact, a file that is written by one program and read by another is a reasonable model of communication over a TCP connection.

When the connection request is accepted, a data socket is created using which the server process can write or read from/to the data stream. When the communication session between the two processes is over, the data socket is closed and the server process is free to accept another connection request. [17]

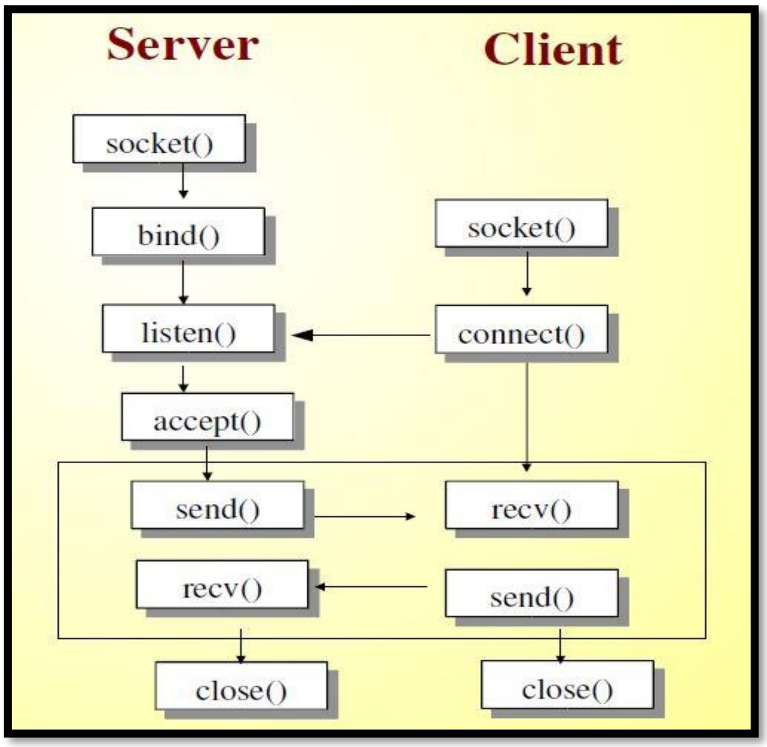


FIGURE 2.3: TCP Socket Algorithm.

No.	Constructor/ Method	Description
		ServerSocket class
1	ServerSocket(int port)	Constructs an object of class ServerSocket and binds the object to the specified port – to which all clients attempt to connect
2	accept( )	This is a blocking method call – the server listens (waits) for any incoming client connection request and cannot proceed further unless contacted by a client. When a client contacts, the method is unblocked and returns a Socket object to the server program to communicate with the client.
3	close( )	Closes the ServerSocket object
4	void setSoTimeout(int timeout)	The ServerSocket object is set to listen for an incoming client request, under a particular invocation of the accept ( ) method on the object, for at most the milliseconds specified in "timeout". When the timeout expires, a java.net.SocketTimeoutException is raised. The timeout value must be > 0; a timeout value of 0 indicates infinite timeout.
		Socket class
5	Socket(InetAddress host, int port)	Creates a stream socket and connects it to the specified port number at the specified IP address
6	InetAddress getInetAddress( )	Returns the IP address at the remote side of the socket
7	InetAddress getLocalAddress( )	Returns the IP address of the local machine to which this socket is bound
8	int getPort( )	Returns the remote port number to which this socket is connected
9	int getLocalPort( )	Returns the local port number to which this socket is bound
10	InputStream getInputStream( )	Returns an input stream for this socket to read data sent from the other end of the connection
11	OutputStream getOutputStream( )	Returns an output stream for this socket to send data to the other end of the connection
12	close( )	Closes this socket
13	void setSoTimeout(int timeout)	Sets a timeout value to block on any read( ) call on the InputStream associated with this socket object. When the timeout expires, a java.net.SocketTimeoutException is raised. The timeout value must be > 0; a timeout value of 0 indicates infinite timeout.

TABLE 2.1: Key Commonly Used Methods of The Stream-Mode Socket API.

2.9.2 The Connectionless Datagram Socket (UDP)

UDP provides an end-to-end service different from that of TCP. In fact, UDP performs only two functions:

1) it adds another layer of addressing (ports) to that of IP

2) it detects some forms of data corruption that may occur in transit and discards any corrupted messages.

Because of this simplicity, UDP sockets have some different characteristics from the TCP sockets. For example, UDP sockets do not have to be connected before being used. [20]

UDP does not attempt to recover from errors experienced by IP; it simply extends the IP best-effort datagram service so that it works between application programs instead of between hosts. Thus, applications that use UDP must be prepared to deal with losses, reordering, and so on. [20]
At the receiving process, a DatagramSocket object must be instantiated and bound to a local port – this port should correspond to the port number carried in the datagram packet of the sender. [17]

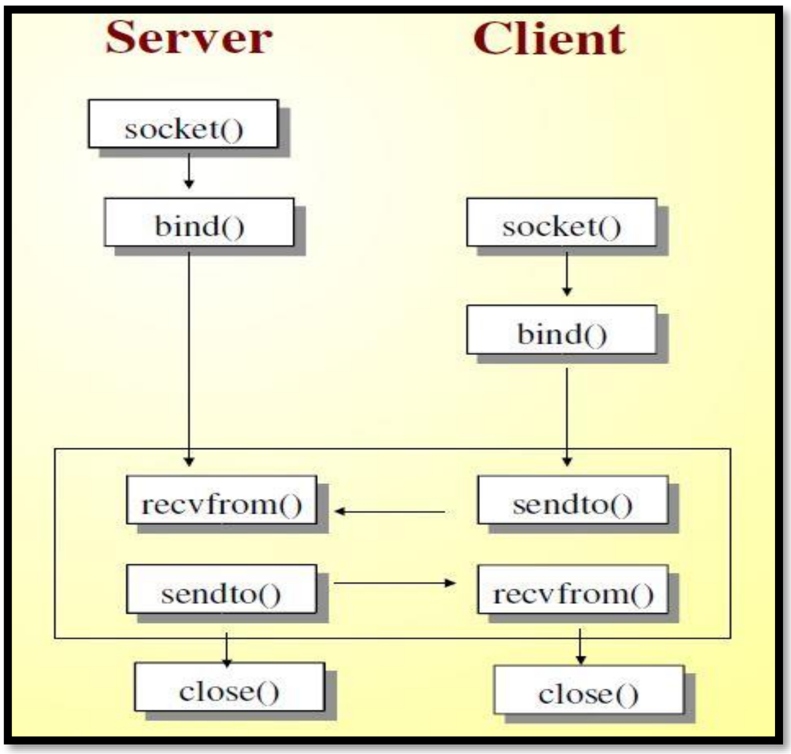


Figure 2.4: UDP Socket Algorithm.

No.	Constructor/ Method	Description
	DatagramSocket class	
1	DatagramSocket( )	Constructs an object of class DatagramSocket and binds the object to any available port on the local host machine
2	DatagramSocket(int port)	Constructs an object of class DatagramSocket and binds it to the specified port on the local host machine
3	DatagramSocket (int port, InetAddress addr)	Constructs an object of class DatagramSocket and binds it to the specified local address and port
4	void connect(InetAddress address, int port)	Connects the datagram socket to the specified remote address and port number on the machine with that address
5	InetAddress getLocalAddress( )	Returns the local InetAddress to which the socket is connected.
6	int getLocalPort( )	Returns the port number on the local host to which the datagram socket is bound
7	void close( )	Closes the datagram socket
8	InetAddress getInetAddress( )	Returns the IP address to which the datagram socket is connected to at the remote side.
9	int getPort( )	Returns the port number at the remote side of the socket
10	void receive(DatagramPacket packet)	Receives a datagram packet object from this socket
11	void send(DatagramPacket packet)	Sends a datagram packet object from this socket
12	void setSoTimeout(int timeout)	Set the timeout value for the socket, in milliseconds
	DatagramPacket class	
13	DatagramPacket(byte[ ] buf, int length, InetAddress, int port)	Constructs a datagram packet object with the contents stored in a byte array.
14	InetAddress getAddress( )	Returns the IP address of the machine at the remote side to which the datagram is being sent or from which the datagram was received
15	byte [ ] getData( )	Returns the data buffer stored in the packet as a byte array
16	int getLength( )	Returns the length of the data buffer in the datagram packet sent or received
17	int getPort( )	Returns the port number to which the datagram socket is bound to which the datagram is being sent or from which the datagram is received
18	void setData(byte [ ])	Sets the data buffer for the datagram packet
19	void setAddress(InetAddress iaddr)	Sets the datagram packet with the IP address of the remote machine to which the packet is being sent
20	void setPort(int port)	Sets the datagram packet with the port number of the datagram socket at the remote host to which the packet is sent

TABLE 2.2: Key Commonly Used Methods of The Datagram socket API.

2.10 Connection modes

The traditional approach of networking is the client-server approach. In this model there is one server talks to arbitrarily many clients to answer their requests. There are two fundamental ways to communicate over a network:

2.10.1 Connected Mode

The connected mode is the most natural way of handling network communications. It corresponds closely to how people are used to communicate say over a phone line. In this mode the two parties share a (virtual) connection that they send data over. The connection first needs to be opened, then data can be sent and eventually the connection is closed. Data sent in such a way has a low chance of getting lost and will always arrive in the order it was sent. Both the traditional telephone network and the TCP for the Internet work in this way.

2.10.2 Disconnected Mode

The disconnected mode is a way of communication in which no connection needs to be established between the parties communicating. In this mode information is broadcast. Since there is no notion of connection it cannot be detected if information has not been picked up by a receiver or information might arrive out of order. Radio and television as well as the UDP for the Internet work in this mode. [9]

2.11 Conclustion

In this chapter, we presented a general introduction of socket APIs by providing some definitions about sockets for more understanding, also we explained the using of addresses. In addition, we focused principally on primary socket procedures and discussed the details of using it in the stream mode and the datagram mode.

In the next chapter, we will present the most important step, which is the cryptography. We will give a global idea about cryptography, we will also discuss the details of its perimeter and the options that make us choose the right method in this work.

Chapter 3

Cryptography

3.1 Introduction

Cryptography is probably the most important aspect of communications security and is becoming increasingly important as a basic building block for computer security.

The increased use of computer and communications systems by industry has increased the risk of theft of proprietary information. Although these threats may require a variety of countermeasures, encryption is a primary method of protecting valuable electronic information. [24]

In this chapter, we will focus on some crucial aspects of cryptography domaine, like general view of classical and modern methods, and cryptography dimensions.

3.2 Basics Of Security And Cryptography

3.2.1 The Perimeter of Cryptography

Most of the time, cryptography is associated with the confidentiality (or privacy) of information only. However, except authorization, it can offer other four functions of security (i.e., authentication, confidentiality, integrity, and nonrepudiation). Let us now see what these terms mean in this context to discuss about the functionalities that cryptography usually has or is supposed to provide. [25]

3.2.1.1 Authentication

It means the process of verification of the identity of the entities that communicate over a network. Without authentication, any user with network access can use readily available tools to forge originating Internet Protocol (IP) addresses and impersonate others. [25]

3.2.1.2 Authorization

It is the process of giving someone permission to do or have something [29]. In other words. Authorization is a security mechanism to determine access levels or user/client privileges related to system resources including files, services, computer programs, data and application features [24].

3.2.1.3 Confidentiality or privacy

It means the assurance that only authorized users can read or use confidential information. Without confidentiality, anyone with network access can use readily available tools to eavesdrop on network traffic and intercept valuable proprietary information. [25]

3.2.1.4 Integrity

It is the property of ensuring correctness in the absence of an actively participating adversary. That sounds more complicated than it really is. What this means in a nutshell is ensuring that a message can be delivered from point A to point B without having the meaning (or content) of the original message change in the process. [26]

3.2.1.5 Nonrepudiation

Nonrepudiation is the property of agreeing to adhere to an obligation. More specifically, it is the inability to refute responsibility. For example, if you take a pen and sign a (legal) contract your signature is a nonrepudiation device. [26]

3.2.2 Security Attacks

A passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter system resources or affect their operation.

3.2.2.1 Passive Attacks

Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are release of message contents and traffic analysis.

3.2.2.2 Active Attacks

Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: masquerade, replay, modification of messages, and denial of service. [24]

3.2.3 Cryptography dimensions

Cryptographic systems are characterized along three independent dimensions:

- The type of operations used for transforming plaintext to ciphertext. All encryption algorithms are based on two general principles: substitution, in which each element in the plaintext (bit, letter, group of bits or letters) is mapped into another element, and transposition, in which elements in the plaintext are rearranged.
- The number of keys used. If both sender and receiver use the same key, the system is referred to as symmetric, single-key, secret-key, or conventional encryption. If the sender and receiver use different keys, the system is referred to as asymmetric, two-key, or public-key encryption.
- The way in which the plaintext is processed (whith the block cipher or the stream cipher).

3.3 Stream Ciphers and Block Ciphers

A stream cipher is a symmetric cipher that operates with a time-varying transformation on individual plaintext digits [27], which encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyed Vigenère cipher and the Vernam cipher.

A block cipher is a method of encrypting text (to produce ciphertext) in which a cryptographic key and algorithm are applied to a block of data (for example, 64 contiguous bits) at once as a group rather than to one bit at a time. [28]

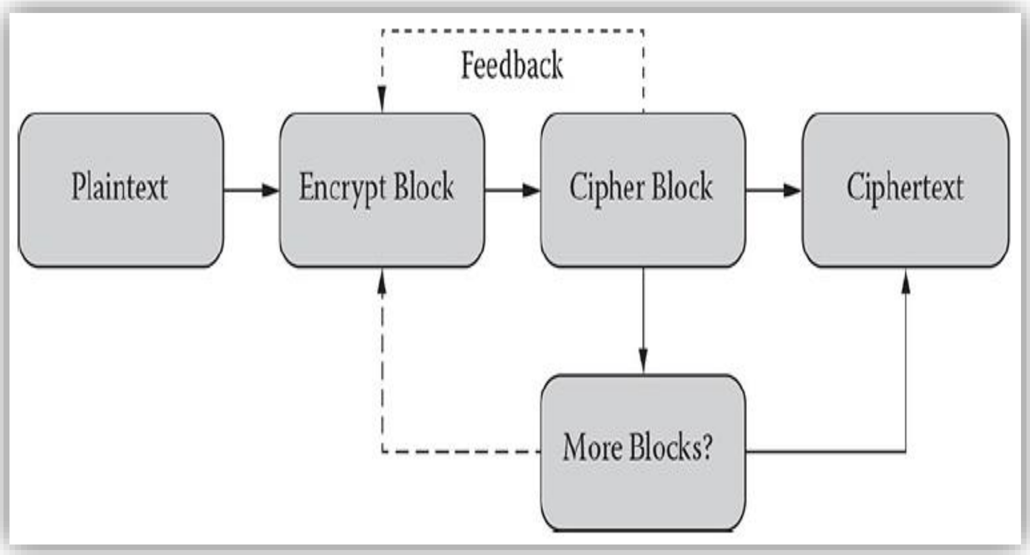


FIGURE 3.1: Sample Operational Diagram of a Block Cipher.

3.4 Classical Cryptographic Algorithms

3.4.1 Caesar Cipher

The earliest known use of a substitution cipher, and the simplest, was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet. [24]

3.4.1.1 Algorithm

- Step 0: Mathematically, map the letters to numbers (i.e., A = 1, B = 2, and so on).
- Step 1: Select an integer key K in between 1 and 25 (i.e., there are total 26 letters in the English language).
- Step 2: The encryption formula is "Add k mod 26"; that is, the original letter L becomes (L + k) mod 26.
- Step 3: The deciphering is "Subtract k mod 26"; that is, the encrypted letter L becomes (L – k) mod 26. [25]

For example, Plaintext–Ciphertext Conversion for Key Value 3 to the Right:

Plain:	**meet me after the toga party**
Cipher:	**PHHW PH DIWHU WKH WRJD SDUWB**

Table 3.1: A Caesar Cipher Conversion for Key Value 3 to the Right

3.4.2 Monoalphabetic Cipher

Monoalphabetic Ciphers With only 25 possible keys, the Caesar cipher is far from secure. A dramatic increase in the key space can be achieved by allowing an arbitrary substitution[24]. Recall the assignment for the Caesar cipher: Unlike Caesar cipher, this technique uses a random key for every single letter (26 keys). [25]

3.4.2.1 Algorithm

- Step 0: Generate plaintext–ciphertext pair by mapping each plaintext letter to a different random ciphertext letter.
- Step 1: To encipher, for each letter in the original text, replace the plaintext letter with a ciphertext letter.
- Step 2: For deciphering, reverse the procedure in step 1. [25]

For example, this is a Sample Plaintext–Ciphertext Letters Mapping in Monoalphabetic Cipher:

Plain:	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**
Cipher:	**Q**	**W**	**E**	**R**	**T**	**Y**	**U**	**I**	**O**	**P**	**A**	**S**	**D**	**F**	**G**	**H**	**J**	**K**	**L**

Table 3.2: A Plaintext Ciphertext Letters Mapping in Monoalphabetic Cipher.

3.4.3 Playfair Cipher

The Playfair cipher was the earliest practical digraph substitution cipher. The technique was invented by Charles Wheatstone in 1854 [25]. It's the best-known multiple-letter encryption cipher is the Playfair, which treats digrams in the plaintext as single units and translates these units into ciphertext digrams. [24]

3.4.3.1 Algorithm

- Step 0: Select the character key. The maximum size of the key is 25, and it can only be letters.
- Step 1: Identify double letters in the key and count them as one.
- Step 2: Set the 5 × 5 matrix by filling the first positions with the key. Fill the rest of the matrix with other letters. J and I will be placed in the same cell as shown in Table 3.3.
- Step 3: Identify double letters in the plaintext and replace the duplicate letter with x (e.g., killer will become kilxer).
- Step 4: Plaintext is encrypted in pairs, two letters at a time. If the plaintext has an odd number of characters, append an x to the end to make it even.
- Step 5: For encryption: (1) If both letters fall in the same row, substitute each with the letter to its right in a circular pattern. (2) If both letters fall in the same column, substitute each letter with the letter below it in a circular pattern. (3) Otherwise, each letter is substituted by the letter in the same row, but in the column of the other letter of the pair.
- Step 6: For deciphering, reverse the procedure in step 5, step 4, and finally, step 3, respectively. [25]

The Playfair algorithm is based on the use of a 5 x 5 matrix of letters constructed using a keyword. Here is an example Sample Playfair Matrix for Key Simple:

S	**I/J**	**M**	**P**	**L**
E	**A**	**B**	**C**	**D**
F	**G**	**H**	**K**	**N**
O	**Q**	**R**	**T**	**U**
V	**W**	**X**	**Y**	**Z**

Table 3.3: Sample Playfair Matrix for Key Simple.

3.4.4 Polyalphabetic Cipher

Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message. The general name for this approach is polyalphabetic substitution cipher. All these techniques have the following features in common:

- A set of related monoalphabetic substitution rules is used.
- A key determines which particular rule is chosen for a given transformation. [24]

A polyalphabetic substitution cipher is a series of simple substitution ciphers. It is used to change each character of the plaintext with a variable length. The Vigenère cipher is a special example of the polyalphabetic cipher. [25]

3.4.4.1 Algorithm

- Step 0: Select a multiple-letter key.
- Step 1: To encrypt, the first letter of the key encrypts the first letter of the plaintext, the second letter of the key encrypts the second letter of the plaintext, and so on.
- Step 2: When all letters of the key are used, start over with the first letter of the key.
- Step 3: The decryption process is the reverse of step 1. The number of letters in the key determines the period of the cipher.

For example, here is a Sample Polyalphabetic Encryption for Key run

Plaintext :	**t**	**o**	**b**	**e**	**o**	**r**	**n**	**o**	**t**	**t**	**o**	**b**	**e**	**t**	**h**	**a**	**t**
Key:	**r**	**u**	**n**	**r**	**u**	**n**	**r**	**u**	**n**	**r**	**u**	**n**	**r**	**u**	**n**	**r**	**u**
Cipher:	**K**	**I**	**O**	**V**	**I**	**E**	**E**	**I**	**G**	**K**	**I**	**O**	**V**	**N**	**U**	**R**	**N**

Table 3.4: Sample Polyalphabetic Encryption for Key Run.

3.5 Data Encryption Standard

3.5.1 Overview

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). For DES, data are encrypted in 64-bit blocks using a 56-bit key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption.

3.5.2 Primitive Operations

All the primitive operations utilized in DES can be separated into two groups: (1) operations for encryption/decryption and (2) operations for key generation. All these operations are discussed below. [25]

3.5.2.1 Operations for Encryption/Decryption

DES encryption/decryption is based on the following primitive operations:

1. Exclusive disjunction/exclusive or (XOR).
2. Initial permutation (IP).
3. Inverse permutation (IP^{-1}).
4. Expansion permutation.
5. Substitution. [25]

3.5.2.2 Operations for Subkey Generation

1. Permuted choice 1 (PC-1).
2. Left shifting.
3. Permuted choice 2 (PC-2). [25]

3.5.3 Modes of operation of the Data Encryption Standard

The Data Encryption Standard (DES) can be implemented into four different modes of operation; two block cipher modes whereas the plaintext data bits are eneiphered as 64-bit data blocks, as well as two stream cipher modes where the plaintext data bits are encrypted individually.

- **Block cipher modes:**
  - ➢ Electronic Codebook Mode (ECB)
  - ➢ Cipher Block Chaining Mode (CBC)
- **Stream cipher modes:**
  - ➢ Output Feedback Mode (OFB)
  - ➢ Cipher Feedback Mode (CFB)

3.5.4 Multiple DES encryption

Multiple DES encipherment provide additionnal cryptographic strength to DES by using more than a single key. In this section, we will see how to protect information using double DES and triple DES.

3.5.4.1 Double DES encryption

Double DES encryption of a plaintext message is achieved by applying the DES encryption transformation on the message with 56-bit key K1 and then applying DES encryption on the resulting 64-bit block with a second 56-bit key K2, The decryption of a ciphertext C from a double DES is obtained by applying the DES decryption transformation twice using first the last encryption key, i.e., K2, and then the first one K1. Figure 3.2 illustrates the double DES encryption and decryption processes.

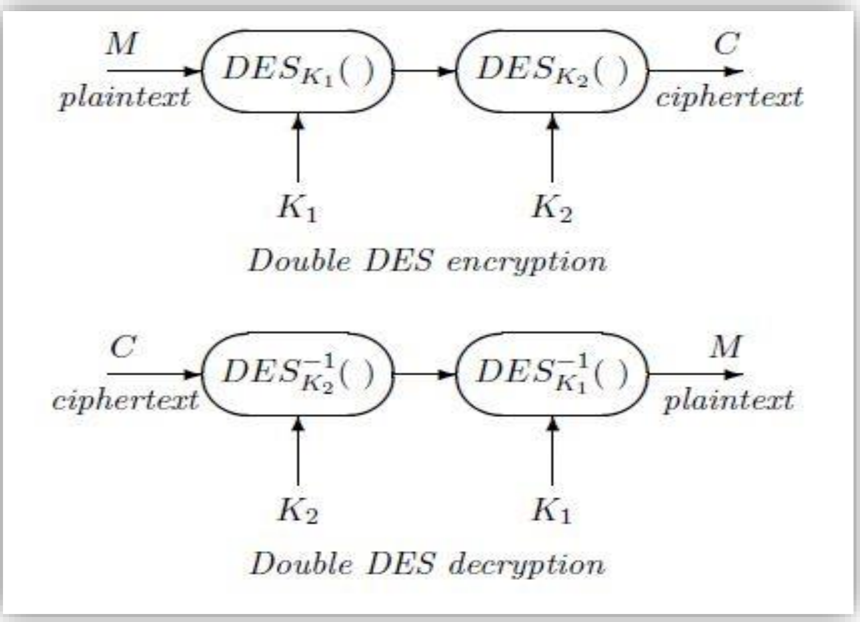


Figure 3.2: Double Des Encryption and Decryption.

3.5.4.2 Triple DES encryption

- **Triple DES encryption with 2 keys**

To prevent a meet in the middle type of attack, a third DES encryption box may be used in cascade with two distinct keys K1 and K2 as shown in Figure 3.3. Triple DES encryption and decryption using two different keys are performed as:

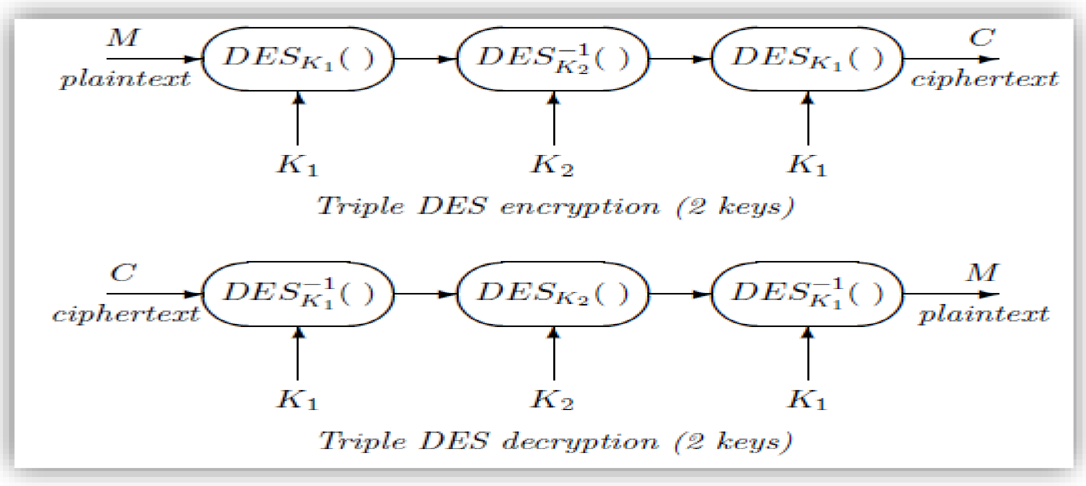


Figure 3.3: Triple Des Encryption and Decryption Using 2 Keys.

- **Triple DES encryption with 3 keys**

Even if there is no known method to break triple DES with two diflerent keys, some still prefer to use triple DES encryption with three different keys (see Figure 3.4 below).

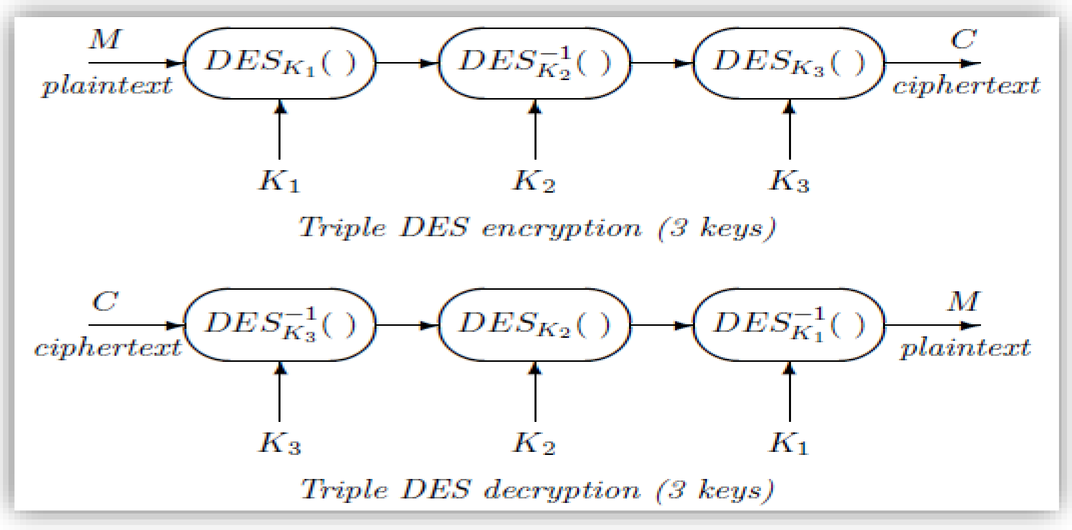


Figure 3.4: Triple DES Encryption and Decryption With 3 Different Keys. [29]

3.6 Advanced Encryption Standard

3.6.1 Overview

The Advanced Encryption Standard (AES) is a renowned symmetric key algorithm that utilizes a same secret key to encrypt and decrypt a message. AES intended to replace DES as the approved standard for a wide range of applications [24]. Moreover, it overcomes the limitation of the smaller key size of the Data Encryption Standard (DES) by utilizing a bigger and variable-length key that may take 149 trillion years to crack (assuming a machine could try 255 keys per second).

In addition, it also resolves the slow processing speed of Triple DES (3DES) and utilizes lower resources than that. AES is now being used worldwide for encrypting digital information, including financial, telecommunications, and government data.

AES supports secret keys of length 128, 192, or 256 bits to encrypt and decrypt a data block of 128 bits. Like other block cipher techniques, it based on permutations and substitutions. Its design supports implementation in both hardware and software. Moreover, it is royalty-free to use, unlike some commercial encryption algorithms. [25]

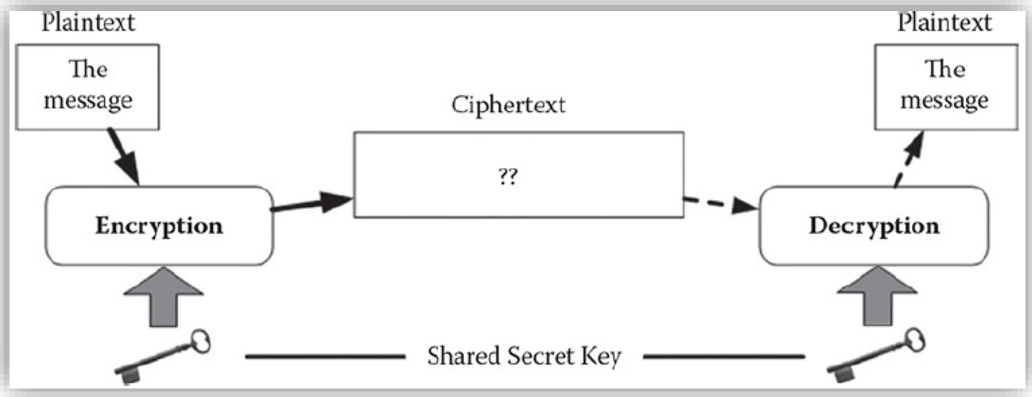


Figure 3.5: Operational Model of Symmetric Cryptography (AES).

3.6.2 History

Because of the limitations of the previous encryption standard (i.e., DES), the NIST was searching for a new symmetric block cipher technique that could be considered a more robust replacement. In the new proposed technique, it was looking for a cipher that could support multiple key sizes (i.e., key lengths), capable of running efficiently in both hardware and software, and also have a good defense mechanism against various attacking techniques.

In this flow of the process, NIST publicly called for nominees for the new algorithm on September 12, 1997. The first AES conference was held from August 20–23, 1998. At that conference, NIST selected 15 candidates for the AES, which were then subjected to preliminary analysis by the world cryptographic community, including the National Security Agency (NSA). All the selected algorithms were presented, analyzed, and tested at the second AES conference, which was held on March 22–23, 1999. On August 9, 1999, NIST selected five algorithms for extensive analysis.

Finally, on October 2, 2000, Rijndael, by Joan Daemen and Vincent Rijmen, was chosen as the Advanced Encryption Standard. On February 28, 2001, the algorithm was included in the publication of a draft by FIPS. Then it was open for public review for 90 days. After that, it was finally included in the Federal Register on December 6, 2001. [25]

3.6.3 The New Standard Requirements

The requirements for the new standard, to be called the Advanced Encryption Standard (AES), were that it should be:

- a 128-bit block cipher with the choice of three key sizes of 128, 192, 256 bits respectively
- a public and flexible design
- at least as secure as two-key triple-DES
- available royalty-free worldwide. [30]

3.6.4 The Basic Structure of Rijndael

Rijndael is a key-iterated block cipher, which alternates key-independent round transformations and key addition. In the basic version (considered here), the cipher encrypts 128-bit blocks in 10 rounds, using 128-bit keys. [30]

The round consists of four different operations, namely, SubBytes, ShiftRows, MixColumn, and AddRoundKey, that are performed repeatedly in a certain sequence; each operation in a standard algorithm maps a 128-bit input state into a 128-bit output state. [30]

3.7 Conclusion

In this chapter, we covered the global meaning of cryptography with a brief presentation about the cryptography perimeter’s. Furthermore, we presented the classical algorithms with mentioning the ciphers types.

This chapter introduces the basic cryptographic concepts and provides background information about the rest of this work when we discuss the modern methods of encryption.

In the next chapter, we will present the conception and implementation that we followed in this work.

Chapter 4

Conception and Implimentation

4.1 Introduction

In the previous chapter, we have seen some related works on cryptography and its methods of encryption used throughout the times. In addition, we discussed the classical cryptographic algorithms like caesar cipher and playfair cipher…etc., and modern methods such as DES and 3DES until access to the latest technology called AES, which came as an alternative to 3DES.

We remind that the objective of our project is to implement an efficient encryption and decryption between the Client and the Server using sockets. In this chapter, we will talk about the AES that we have chosen in our work, for its strength and efficiency compared to other techniques, and explain how it works in detail with mention of its benefits. Moreover, we will talk about the client/server software and its mechanisms that will be taken to achieve our goal.

4.2 Conception

4.2.1 General Conception

Client/server is a software architecture model consisting of two parts, client system and server system, both communicating over a computer network or on the same computer.

Users use open communication channels for data transfer. Therefore, data transfer over such channels is in extreme need of protection in order to save confidentiality. One of the most necessary steps on data protection is the data encryption. Encryption is the process of transforming the data into some sequence of bytes using one of encryption algorithms.

The primary goal of encryption is to hide the data from being visible and accessible without having a key. We use the secret key cryptography (The symmetric cryptography).

Figure 3.1 presents the proposed basic architecture for the client/server software. The server remains listen and waiting for the client to contact it. When the client connects to the server, he can send files of all types and sizes one by one to be received by the server and will be stored.

The client encrypts the file then will sends it over the network to the server, which will decrypt it there.

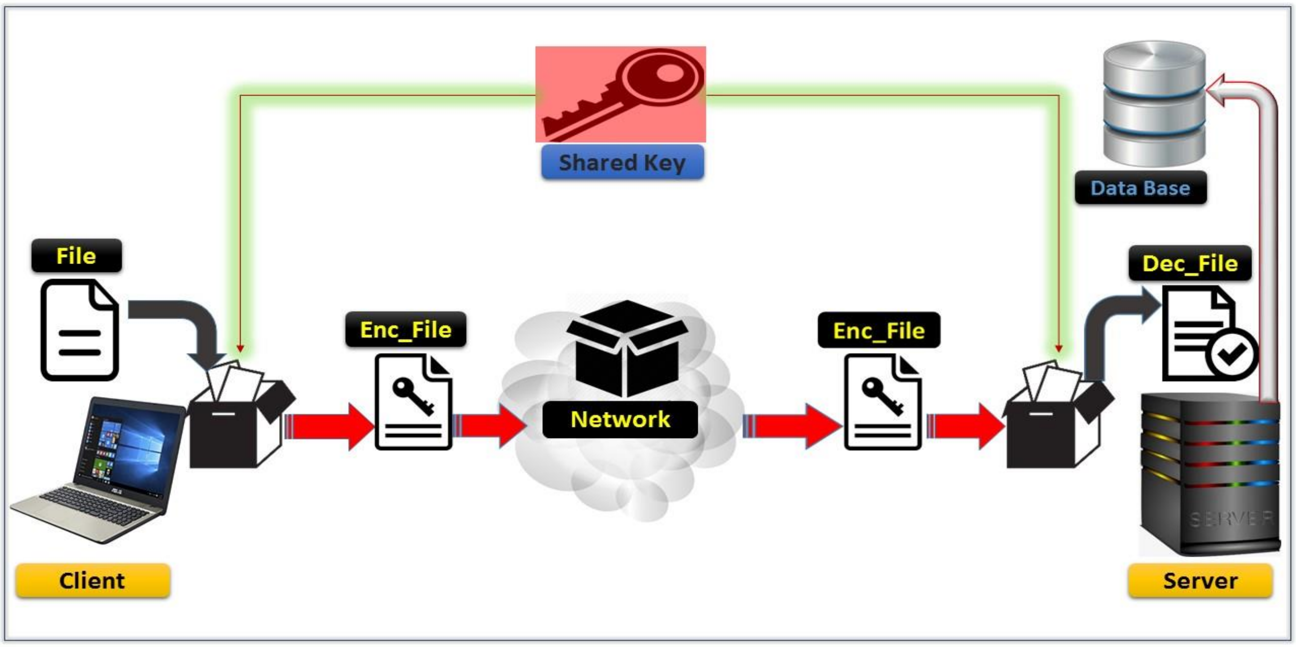


Figure 4.1: Client/Server Global Architecture.

4.2.2 Detailed Conception

In this section, we will discuss the choices that make this program effective and well protected. We focus on three importants parts, which are the design of the application, the transfer of data, and the encryption of data.

4.2.2.1 The Two-tiers client/server architecture

Our client/server application based on two-tier architecture. Here are the most important benefits:

- The direct communication takes place between client and server.
- There is no intermediate between client and server.
- Communication is faster.
- Easy to implement and optimize performance.

4.2.2.2 The connected mode

Our case is about ensuring the delivery of information, which may be documents, pictures, applications, etc. for this we choose the connected mode.

In the TCP mode, the two network devices are directly connected, the two sides establish a virtual link before the actual data transfer. An error message is returned if the information is not correctly received. TCP sorts and collects received packets. If a package is missing or defective, the receiver will request the resend of the information.

The connected mode assuring reliability because data transfer is synchronized between the two devices that check if the data sent is actually received.

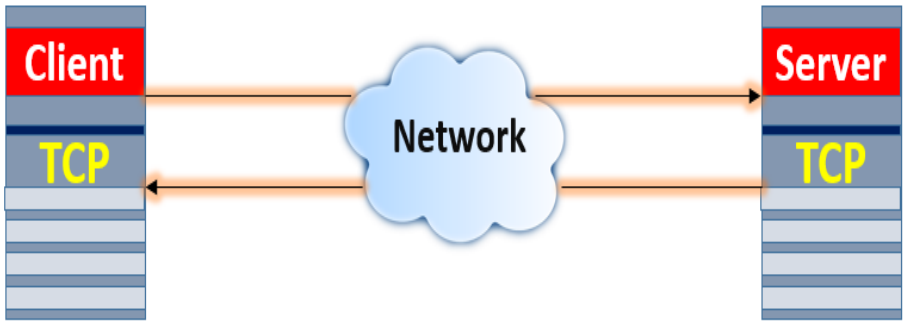


Figure 4.2: Simple Design of Sockets Connected Mode.

4.2.2.3 Secure of Data Using AES Encryption

AES is a symmetric encryption algorithm. It uses the same key for encryption and decryption. Large amounts of data can be encrypted using a symmetric encryption algorithm. Here we will discuss about how it works, its design, and some detailed informations about it.

a) Design Consideration of AES

One of the principal design goals of AES was to keep it simpler to implement in both hardware and software. Therefore, unlike DES, instead of operating on bits, it operates on bytes, which makes it easier to implement and explain.

It works by repeating the same defined steps multiple times, which are called rounds. Each round consists of several processing steps, including one that utilizes an encryption/decryption subkey that is generated from the shared key.

Since AES is an iterative symmetric block cipher, it shares a single secret key between the two communicating parties involved in encryption and decryption operations. The allowable key lengths in AES are 128, 192, and 256 bits. Every key is expanded so that a separate subkey (w [i, j], where i and j provide the byte range) could be utilized for every round. [34]

Number of rounds of AES generally depends on the key length. A relation between key length, number of columns in a state, and number of rounds is mentioned in Table 4.1. For instance, if the key length (N_k) is 128 bits or 16 bytes or 4 words, the number of columns (N_b) would be 4 and only 10 rounds (N_r) are performed, where N_b = key length/32. [25]

Key Length (N_k)	Number Of Columns In State (N_b)	Rounds (N_r)
128 bits	4	10
192 bits	6	12
256 bits	8	14

Table 4.1: Relation between N_k, N_b, and N_r.

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Figure 4.3: A State of 128-Bit Key AES.

AES, as well as most of the encryption algorithms, is reversible, which means that for the steps performed to complete an encryption, similar steps could be followed to complete a decryption, but in reverse order. In the following section, a detailed description of the operations of AES is explained with examples.

b) Primitive Operations of AES

Internally, all the AES operations are performed on a two dimensional array of bytes called the state. A state constitutes four rows and N_b (Table 4.1) number of columns. Hence, for a 128-bit key, a state consists of four rows and four columns, as depicted in Figure 4.3. AES is based on the next five primitive operations.

b).1 Exclusive disjunction or exclusive OR (XOR)

XOR is a logical operation that outputs true whenever both inputs differ from each other (e.g., one is true and the other is false) (Table 4.2). [31]

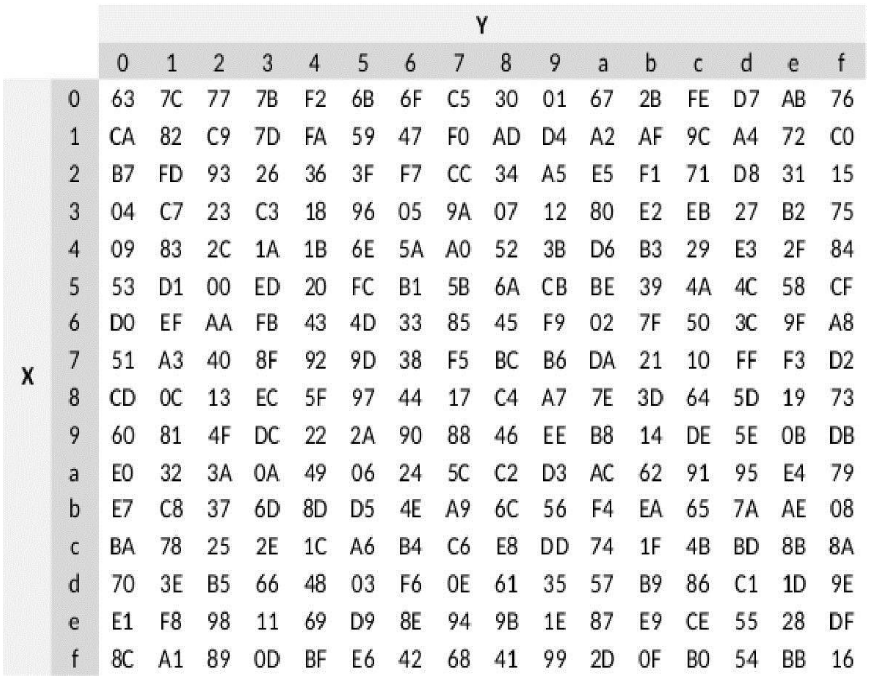
Input		Output
0	0	0
0	1	1
1	0	1
1	1	1

Table 4.2: XOR Truth Table.

b).2 Substitution (SubByte)

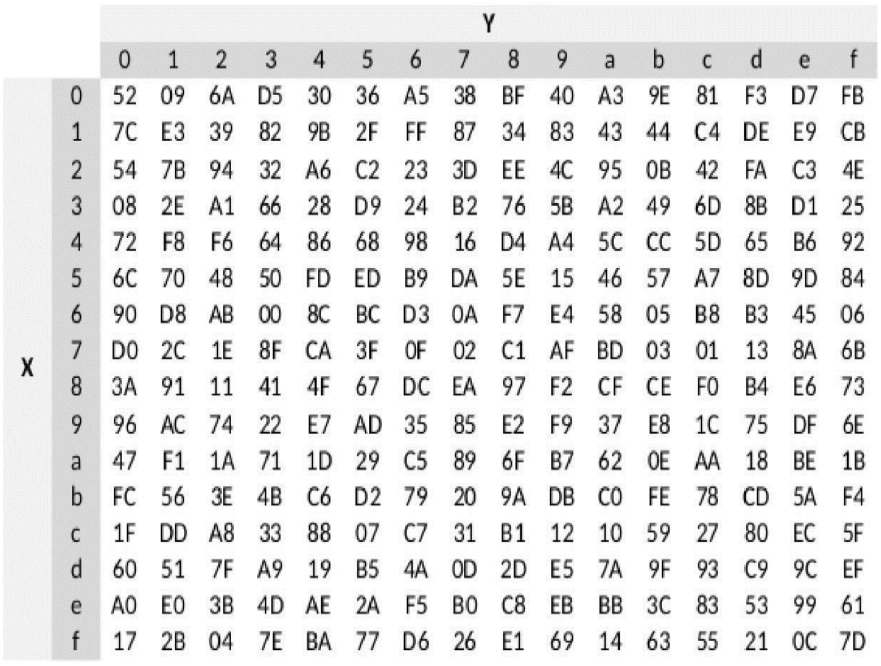
A byte is substituted by another byte. AES utilizes a lookup table, also known as S-box, to perform substitutions of encryption, and another S-box, also known as inverse S-box, for decryption[32]. Both S-boxes are given in Tables 4.3 and 4.4, respectively.

Each individual byte can be represented by two hex digits where the first (from right) digit represents row and the second digit represents column of the S-box lookup table in the case of encryption, and of the inverse S-box in the case of decryption. For instance, let us assume that {68} is a hexadecimal value that represents a byte. Here, 6 refers to row number and 8 refers to column number; the value over that location would substitute this value {45}. [25]



		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
X	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	a	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	b	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	c	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	d	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	e	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	f	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Table 4.3: S-Box Lookup Table.



		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	83	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
X	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	a	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	b	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	c	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	d	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	e	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	f	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Table 4.4: Inverse S-Box Lookup Table.

b).3 Rotation (ShiftRows)

A simple permutation is performed by rearranging of bytes through rotating a row by a fixed number of cells. It provides a diffusion by the cyclic left shift of the last three rows of the state by different offsets. Row 0 of the state is not shifted, row 1 is shifted 1 byte, row 2 is shifted 2 bytes, and row 3 is shifted 3 bytes. This operation is illustrated in (Figure 4.4). [31]

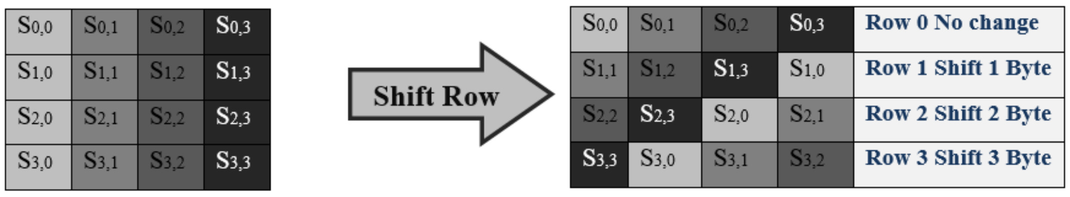


Figure 4.4: Shift Row Operation of AES.

In case of decryption, inverse shift rows (InvShiftRows) are performed, which follows a process similar to that of ShiftRows, only the shifting is done to the right.

b).4 MixColumn

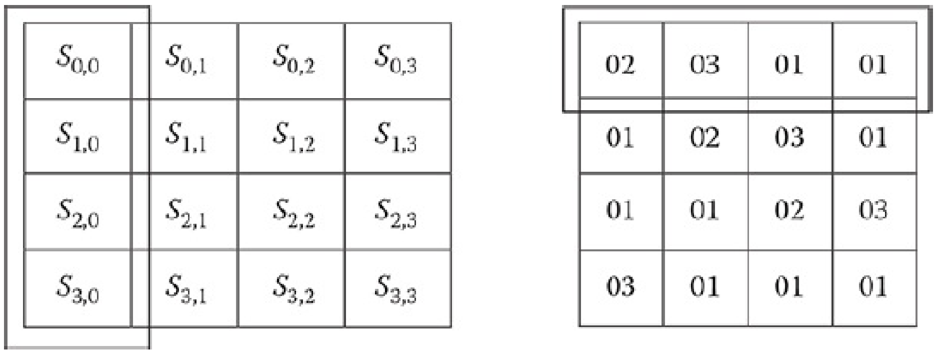
It operates on each column individually where a single byte of a column is mapped into a new value that is a function of all four bytes in that column [31] . Each column of the state is replaced by multiplying with a 4 × N_b matrix in the Galois field 2 , also denoted as GF(2). The first result byte is calculated by multiplying four values of the state column against four values of the first row of the matrix. The result of each multiplication is then XORed to produce 1 byte like below [33]:

$$\mathbf{S_{0,0} = (S_{0,0}*2)\ XOR\ (S_{1,0}*3)\ XOR\ (S_{2,0}*1)\ XOR\ (S_{3,0}*1)}$$

This procedure is repeated again with each byte of all columns of the state, until there is no more state column. As a result of this multiplication, the four bytes in the first column are replaced by the following:

$$\mathbf{S_{0,0} = (S_{0,0}*2)\ XOR\ (S_{1,0}*3)\ XOR\ (S_{2,0}*1)\ XOR\ (S_{3,0}*1)}$$
$$\mathbf{S_{1,0} = (S_{0,0}*1)\ XOR\ (S_{1,0}*2)\ XOR\ (S_{2,0}*3)\ XOR\ (S_{3,0}*1)}$$
$$\mathbf{S_{2,0} = (S_{0,0}*1)\ XOR\ (S_{1,0}*1)\ XOR\ (S_{2,0}*2)\ XOR\ (S_{3,0}*3)}$$
$$\mathbf{S_{3,0} = (S_{0,0}*3)\ XOR\ (S_{1,0}*1)\ XOR\ (S_{2,0}*1)\ XOR\ (S_{3,0}*2)}$$

An example of MixColumn is given for 128-bit key in (Figure 4.5).



$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
02 03 01 01
01 02 03 01
01 01 02 03
03 01 01 01

Figure 4.5: 128-Bit Key State And Its Multiplication Matrix.

b).5 AddRoundKey

This is a simple operation where each byte of the state is XORed with each byte of the round key, which is a portion of the expanded key. In the next section, a detailed description of the key expansion technique of AES is elaborated. Figure 4.6 illustrates the technique of AddRoundKey transformation. [25]

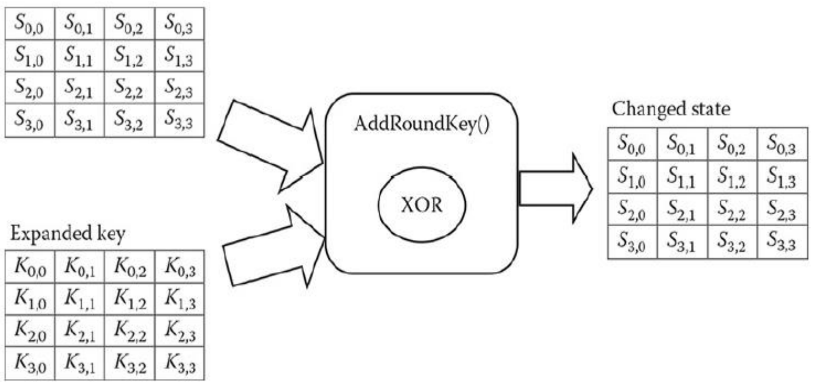


Figure 4.6: Add Round Key Function.

c) Structure of AES

The basic encryption and decryption structure of AES is illustrated in Figure 4.7. Here, a 128-bit key length is considered. Therefore, both encryption and decryption must go through 10 rounds before producing the desired output. There are 12 rounds for a 192-bit key and 14 rounds for a 256-bit key.

It can be observed from the figure that every round generally performs four operations: (1) SubBytes/InvSubBytes, (2) ShiftRows/InvShiftRows, (3) MixColumns/InvMixColumns, and (4) AddRoundKey. One of them is permutation and the other three are substitutions. However, the final round comprises only three operations, excluding MixColumns/InvMixColumns. [32]

The expanded key is only utilized by the AddRoundKey operations. Each operation is easily reversible, thus making it easy to implement in both hardware and software. Similar to most of the block ciphers, the decryption algorithm utilizes the key in reverse order. [34]

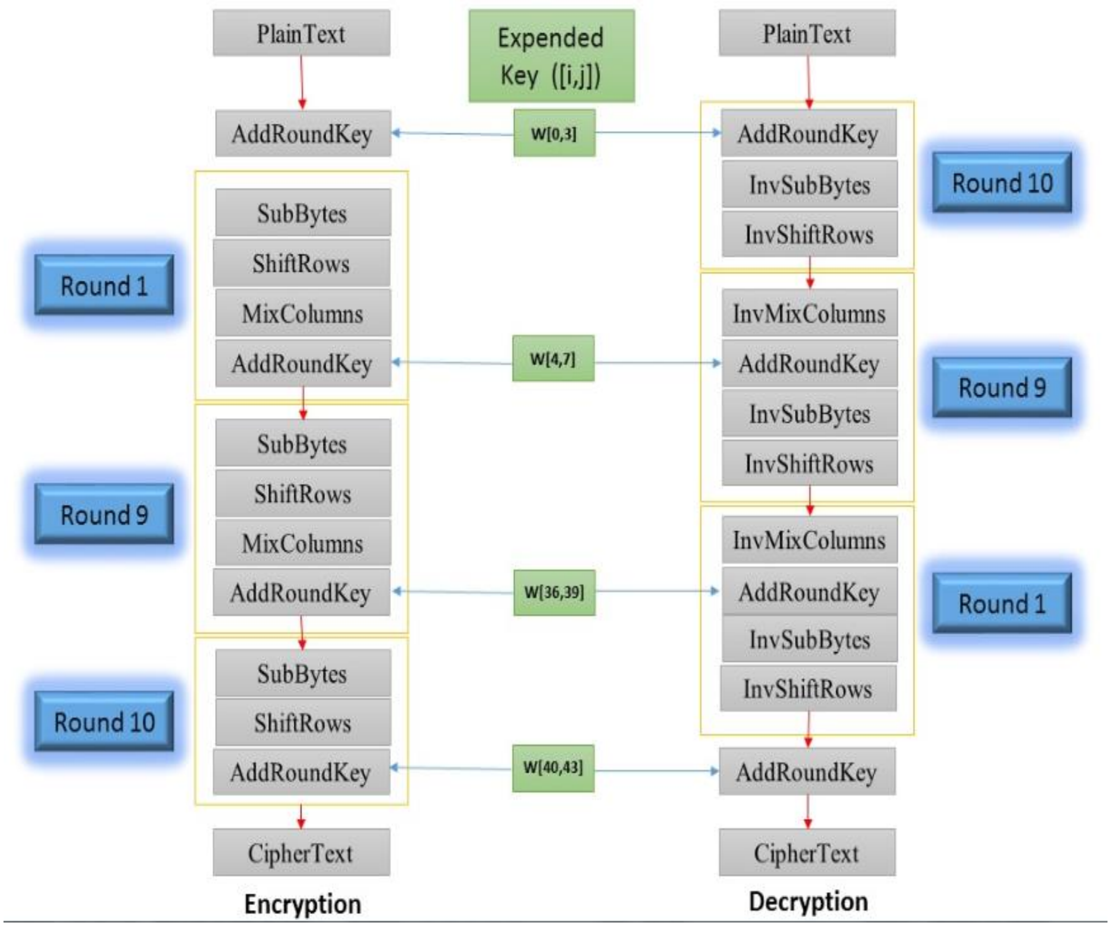


Figure 4.7: AES Encryption and Decryption Techniques.

d) Overview of Key Expansion

As mentioned earlier, since AES supports symmetric key [33], a secret key must be shared between the two parties. AES provides flexibility regarding selecting a key length. A key could be 128, 192, or 256 bits long. Since every round utilizes a new subkey, prior to encryption or decryption, the key must be expanded according to the number of rounds. This process is called key expansion. [25]

4.3 Implementation

In order to achieve our goal, which is the conception of a client/server system using cryptography to encrypt exchanging data, we use windows as an operating system and java as a platform for programming, and wireshark to check communication results.

In this section, we will explain all the steps that we have took to implement the client/server system whith crypted information exchange (text, image…), and all the results obtained will be presented and analyzed.

4.3.1 Tools and Language Programming Used

4.3.1.1 Java

Java is one of the most popular programming languages used to create applications and platforms. It was designed for flexibility, allowing us to write code that would run on any machine, regardless of architecture or platform. Java gave the ability to move easily from one computer system to another.

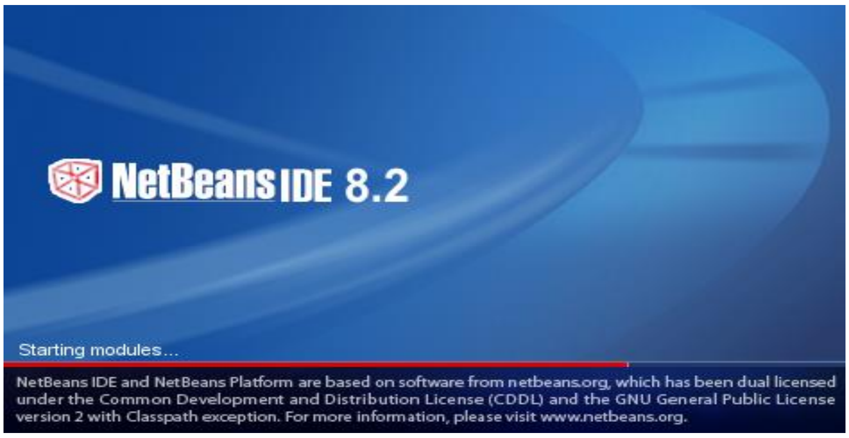
Java applications are able to manage their own use by multiple users at the same time, creating threads for each use within the program itself, rather than having to run multiple copies of the programming in the same hardware.

Java was designed also to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages. This allow us to create modular program and reusable code. [35]

4.3.1.2 NetBeans

NetBeans is an open-source integrated development environment (IDE) for developing with Java, PHP, C++, and other programming languages. NetBeans is also referred as a platform of modular components used for developing Java desktop applications.

NetBeans IDE is the official IDE for Java 8. With its editors, code analyzers, and converters, we can quickly and smoothly upgrade our applications to use new Java 8 language constructs, such as lambdas, functional operations, and method references. [36]



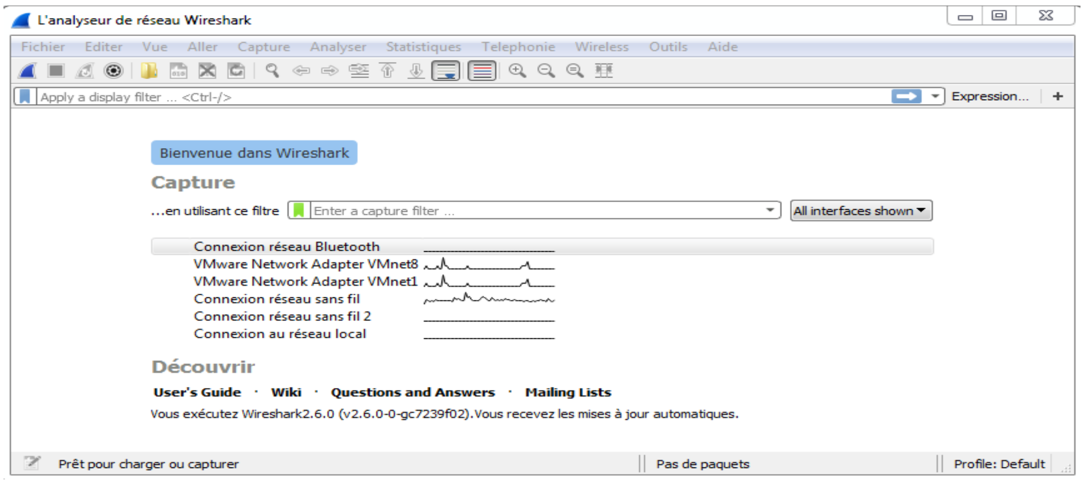
NetBeans IDE and NetBeans Platform are based on software from netbeans.org, which has been dual licensed under the Common Development and Distribution License (CDDL) and the GNU General Public License version 2 with Classpath exception. For more information, please visit www.netbeans.org.

Figure 4.8: Netbeans

4.3.1.3 Wireshark

Wireshark is a network packet analyzer. It will try to capture network packets and tries to display that packet data as detailed as possible. In other words, we could think of a network packet analyzer as a measuring device used to examine what's going on inside a network cable,

In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, all that has changed.Wireshark is perhaps one of the best open source packet analyzers available today. [37]


L'analyseur de réseau Wireshark
Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide
Apply a display filter ... <Ctrl-/>
Expression...
+
Bienvenue dans Wireshark
Capture
Enter a capture filter ...
All interfaces shown
Découvrir
User's Guide · Wiki · Questions and Answers · Mailing Lists

...en utilisant ce filtre

Connexion réseau Bluetooth
VMware Network Adapter VMnet8
VMware Network Adapter VMnet1
Connexion réseau sans fil
Connexion réseau sans fil 2
Connexion au réseau local

Vous exécutez Wireshark2.6.0 (v2.6.0-0-gc7239f02).Vous recevez les mises à jour automatiques.

Prêt pour charger ou capturer | Pas de paquets | Profile: Default

Figure 4.9: Wireshark.

4.3.2 Client/server application using AES encryption

The program in general works to send encrypted files from the client to the server and uses AES encryption. We will explain the software design, how it works, and all the steps in detail separately.

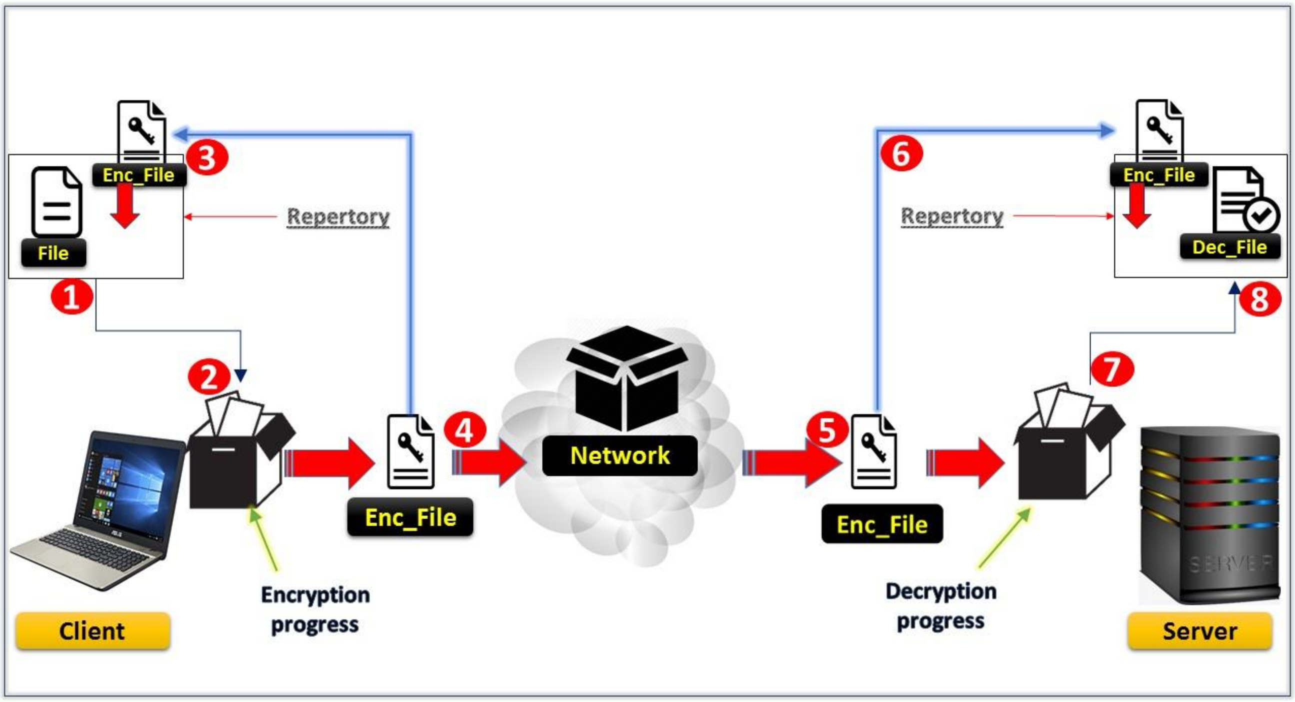


Figure 4.10: Most Important Client/Server Application Operations.

The numbers in (Figure 4.10) refers to the next actions:

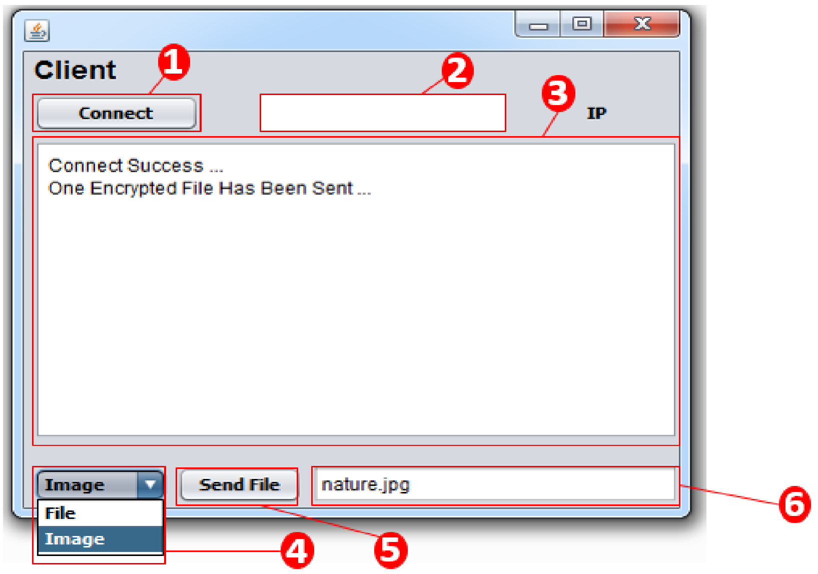
1- The client choose a file.
2- The client encrypts the file.
3- The client places a copy of the encrypted file in repertory.
4- The client sends the encrypted file over the network.
5- The server receives the encrypted file.
6- The server places a copy of the encrypted file in the repertory.
7- The server decrypts the file.
8- The server saves the decrypted file in repertory.

4.3.2.1 Client side (Encryption Process)

This program is designed to works as a client that sends encrypted files to the server. After the connection between the client and the server and after the client confirmed its acceptance, the client first chooses and encrypts the file whith the AES algorithm. After that, the client places an encrypted copy of the file in the same place as the original file, the client then sends the encrypted file to the server, which in turn receives it and stores it.

The client can connect to the server if they are on the same device and this is when no value entered for the IP field. If the client wants to connect to the server whene they being on two differents devices, it must enter the IP address of the server to communicate. The next Figure 4.11 illustrates the client graphical interface.

a) Graphical user interface



Connect Success ...
One Encrypted File Has Been Sent ...

Figure 4.11: The Client (Graphical Interface).

The numbers in (Figure 4.11) refers to:

1- A button to connect the server.
2- An area to put the ip address (we keep it empty if we want to connect to localhost).
3- An area shows the messages updates.
4- A button to change the shape between Image and File.
5- A button to choose a file and send it.
6- An area to naming the file that we want to send.

b) Algorithm

The primary algorithms that we use in our client application are divided into three basic components, which about:

1- Sockets

- Creates a stream socket and connects it to the specified port number at the specified IP address.

• **Socket(InetAddress host, int port)**

2- paths and dispatchs

- Getting the main file path and determining the new file path after generating an encrypted file.

• **String=file.substring()**
• **File = new File()**

- Generating the streams channel that read from main file in a buffer and encrypt it then write from the buffer to the original file path.

• **InputStream = new FileInputStream()**
• **OutputStream = new FileOutputStream()**
• **CipherInputStream = new CipherInputStream();**

3- Encryption

- Converting the String AES key To a SecretKey Object to be used in encryption.

• **byte[] decodedKey = Base64.getDecoder().decode()**
• **SecretKey secretKey = new SecretKeySpec()**

- Generating a cipher object responsible for encryption process steps for AES Algorithm.

• **Cipher cipher = Cipher.getInstance("AES")**
• **cipher.init(Cipher.ENCRYPT_MODE, secretKey)**

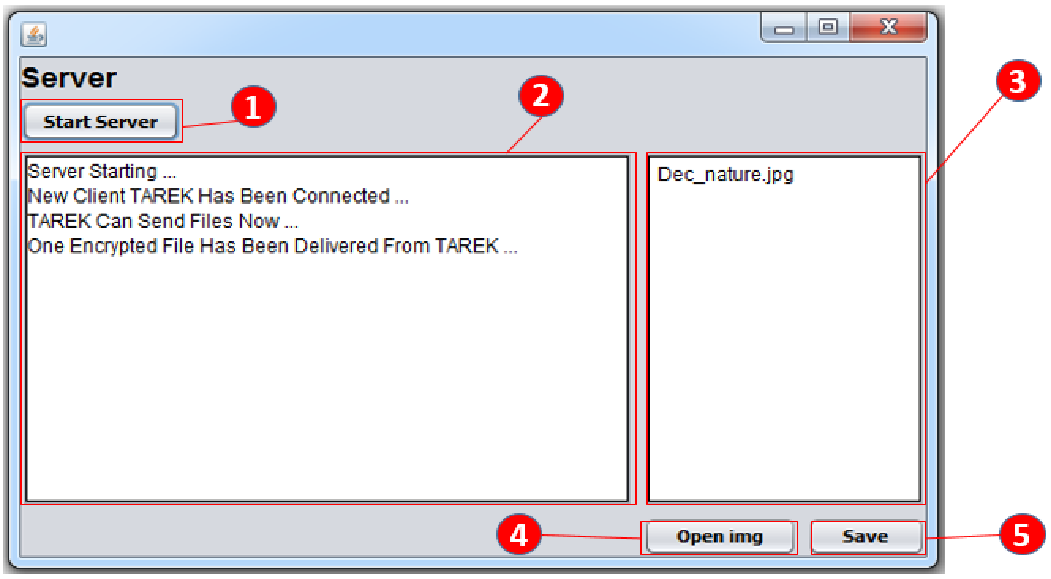
4.3.2.2 Server side (Decryption Process)

Server Side has been designed to receive files from the client, the server must open a connection first and wait for the client to contact it.

After the client accepteance, the server will be ready to receive the encrypted files from the client to decrypt them. a copy of the encrypted file and the decrypted file will be in the same server location.

The server can save the file after the decryption anywhere he want, and he have the possibility to open it from the graphical interface if this file is an image.

a) Graphical user interface


Server Starting ...
New Client TAREK Has Been Connected ...
TAREK Can Send Files Now ...
One Encrypted File Has Been Delivered From TAREK ...

Dec_nature.jpg

Figure 4.12: : The Server (Graphical Interface).

The numbers in (Figure 4.12) refers to:

1- A button starts the server and turns it to state of listening

2- An area for messeges updates

3- An area shows a liste of delivered file names

4- A button to open an image

5- A button to save the files anywhere

b) Algorithm

The primary algorithms that we use in our server application are divided into 3 basic components, which about:

1- sockets

- Constructs an object of class ServerSocket and binds the object to the specified port – to which all clients attempt to connect.

  - **ServerSocket(int port)**

- The server listens (waits) for any incoming client connection request and cannot proceed further unless contacted by a client.

  - **accept( )**

2- paths and dispaths

- Getting the main file path and the regenerated new file path for decrypting the encrypted file.

  - **File src = new File()**

- Generating the streams channel that read from main file in a buffer and decrypt it then write from the buffer to the decrypted file path.

  - **InputStream = new FileInputStream()**
  - **OutputStream = new FileOutputStream()**
  - **CipherOutputStream = new CipherOutputStream()**

3- Decryption

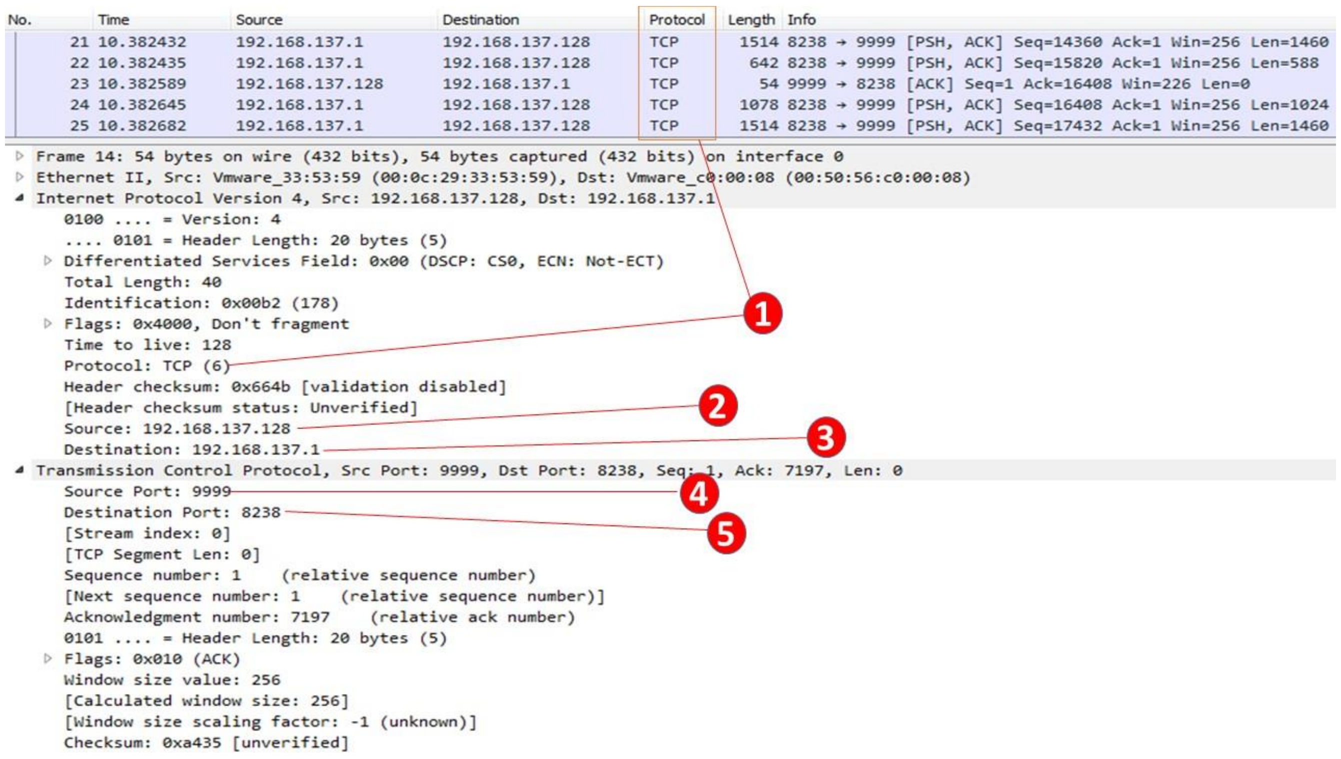
- Converting the String AES key To a SecretKey Object to be used in decryption.

  - **byte[] decodedKey = Base64.getDecoder().decode()**
  - **SecretKey secretKey = new SecretKeySpec()**

- Generating a cipher object responsible for decryption process steps for AES Algorithm.

  - **Cipher cipher = Cipher.getInstance("AES")**
  - **cipher.init(Cipher.DECRYPT_MODE,secretKey)**

4.3.3 Checking Connectivety



```
No.      Time       Source           Destination      Protocol  Length Info
      21 10.382432  192.168.137.1    192.168.137.128  TCP       1514 8238 → 9999 [PSH, ACK] Seq=14360 Ack=1 Win=256 Len=1460
      22 10.382435  192.168.137.1    192.168.137.128  TCP        642 8238 → 9999 [PSH, ACK] Seq=15820 Ack=1 Win=256 Len=588
      23 10.382589  192.168.137.128  192.168.137.1    TCP         54 9999 → 8238 [ACK] Seq=1 Ack=16408 Win=226 Len=0
      24 10.382645  192.168.137.1    192.168.137.128  TCP       1078 8238 → 9999 [PSH, ACK] Seq=16408 Ack=1 Win=256 Len=1024
      25 10.382682  192.168.137.1    192.168.137.128  TCP       1514 8238 → 9999 [PSH, ACK] Seq=17432 Ack=1 Win=256 Len=1460

▷ Frame 14: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
▷ Ethernet II, Src: Vmware_33:53:59 (00:0c:29:33:53:59), Dst: Vmware_c0:00:08 (00:50:56:c0:00:08)
◢ Internet Protocol Version 4, Src: 192.168.137.128, Dst: 192.168.137.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▷ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x00b2 (178)
  ▷ Flags: 0x4000, Don't fragment
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x664b [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.137.128
    Destination: 192.168.137.1
◢ Transmission Control Protocol, Src Port: 9999, Dst Port: 8238, Seq: 1, Ack: 7197, Len: 0
    Source Port: 9999
    Destination Port: 8238
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 7197    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  ▷ Flags: 0x010 (ACK)
    Window size value: 256
    [Calculated window size: 256]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xa435 [unverified]
```

Figure 4.13: Checking Connectivety whith Wireshark.

The numbers in (Figure 4.13) refers to:

1- Here the Wireshark analysis shows that the application uses the connected mode (TCP).

2- The Ip address of the server is (192.168.137.128).

3- The Ip address of the client is (192.168.137.1).

4- The port number of the server is (9999).

5- The port number of the client is (8238).

4.4 Conclusion

In this chapter, we divided this work into two major titles. In addition to the description of the steps that we followed to create an efficient application that can encrypt exchanging data between the client and the server. As a first step, we presented the global architecture of this work. Then, we explained all the options that we took to implement our application.

After that, we started the implementation; this section focused on how we can build the application and tools that are used. Then we explained the graphical interfaces in details. We end the chapter with checking of the connectivity and we confirmed that the application used the connected mode.

General Conclusion

The importance of computers and networks and the information they store and communicate to society today are equaled only by the threats to them. As threats increase, solutions evolve, new ways emerge, and the cryptography evolution provides the AES algorithm as a great solution.

The Advanced Encryption Standard is one of the most widely used ciphers for the encryption of data. It has been approved by the United States government to protect classified data. AES performs very quickly and its simplicity means that operations can be performed very fast, and is easily implemented on hardware and software. All of these properties are why Rijndael was chosen by NIST to become AES.

Symmetric-key cryptography is characterized by the use of a single secret key to encrypt and decrypt secret information. This use of a single key is where the name symmetric came from, the same algorithm and key are used in both directions. Symmetric key cryptography finds a welcome home in the client/server model. However, when we used the AES algorithm, we saw that it was very effective and the encryption and decryption was very fast regardless of files size and type.

The connectivity results reached in Wireshark show that the application was worked on the connected mode and show ports and ip addresses that the client/server application used.

Bibliography

[1] R. Orfali , J.Edwards. D. Harkey, ***“Client/Server Survival Guide’’***, John Wiley and Sons Ltd , New York, United States, 08 Feb 1999.

[2] J. R. Mariga, J. E. Goldman , Ph. T. Rawles, ***“Client/Server Information Systems : A Business-Oriented Approach’’***, John Wiley & Sons Inc , New York, United States, 31 Dec 1998.

[3] A. Schill, ***“Distributed Platforms’’***, Springer-Verlag New York Inc , New York, NY, United States, 09 Mar 2013.

[4] D. E. COMER, ***“Computer Networks and Internets’’***, Upper Saddle River , New Jersey 07458, 2009.

[5] B. Souheyla, ***“Etude et Administration des Systèmes de Supervision dans un Réseau Local’’***, Master’s Thesis In Science, University of Abou Bakr Belkaid Tlemcen, Department Of Science, 2011.

[6] S. Ch. Yadav, S. K. Singh, ***“Introduction to Client/Server Computing’’***, New Age International Pvt Ltd Publishers , Ansari Road, Daryaganj, New Delhi, 2009.

[7] C. Rendell, ***“Network Topologies : Types, Performance Impact & Advantages / Disadvantages’’***, Nova Science Publishers Inc , New York, United States, 01 Jul 2013.

[8] I. J. Taylor, A. Harrison, ***“From P2P to Web Services and Grids,Peers in a Client/Server World’’***, Springer Verlag London Limited, Cardiff, Wales, 2005.

[9] B. Ciubotaru, G. Muntean, ***“Advanced Network Programming : Principles and Techniques : Network Application Programming with Java’’***, Springer London Ltd , United Kingdom, 07 Aug 2015.

[10] C. M. Kozierok, ***"The Tcp/ip Guide"***, No Starch Press , San Francisco, United States,US, 30 Nov 2005.

[11] D. A. Milovanovic, Z. S. Bojkovic ,K. R. Rao, ***"Introduction to Multimedia Communications : Applications, Middleware"***, John Wiley and Sons Ltd , Networking, Chichester, United Kingdom, 20 Jan 2006.

[12] L. Chung, ***"Client-Server Architecture"***, Dallas: Computer Science Program, The University of Texas.

[13] R. M. Newman, E. Gaura , D.Hibbs, ***"Computer Systems Architecture"***, Lexden Publishing Limited, Colchester, United Kingdom, 31 May 2008.

[14] B. Fenner, A. M. Rudoff, W. R. Stevens, ***"Unix Network Programming, Volume 1 : The Sockets Networking API"***, Pearson Education, New Jersey, United States ,24 Nov 2003.

[15] Web Site for pace university, ***"http://csis.pace.edu/~marchese/CS865/Lectures/Liu4/sockets"***, visited in 16 Avr 2018.

[16] James C. Foster, Mike Price, ***"Sockets Shellcode Porting And Coding Reverse Engineering Exploits And Tool Coding For Security Professionals"***, Syngress Publishing, Inc ,Hingham Street, Rockland, MA 02370,USA, 2005.

[17] N. Meghanathan, ***"A Tutorial on Socket Programming in Java"***, Jackson State University ,Jackson, MS 39217, USA, 2010.

[18] C. Marxer, ***"Socket Programming in C, Introduction to Internet and Security"***, University of Basel ,Switzerland.

[19] D. Szameitat, ***"Client-Server-Socket Programmierung : Unterstutzung in Unterschiedlichen Programmiersprachen"***, GRIN Publishing ,Munich, Germany, 23 Aug 2013.

[20] Kenneth L. Calvert, Michael J. Donahoo, ***"TCP/IP Sockets in C, Pratical Guide for Programmers"***, Elsevier Science & Technology ,San Francisco, United States, 07 May 2009.

[21] Qing Li, Keiichi Shima , Jinmei Tatuya, ***“IPv6 Socket API Extensions: Programmer's Guide’’***, Elsevier Science & Technology ,San Francisco, United States, 01 Oct 2009.

[22] E. R. Harold, ***“Java Network Programming’’***, O'Reilly Media, Inc ,Sebastopol, United States, 17 Oct 2013.

[23] M. Jones, ***“BSD Sockets Programming From a Multi-Language Perspective’’***, Cengage Learning Inc, Hingham, United States, 01 Oct 2003.

[24] W. Stallings, ***“Cryptography and Network Security : Principles and Practice: fourth edition’’***, Prentice Hall ,New Jersey, United States, November 16, 2005.

[25] S. Azad, A. K. Pathan, ***“Practical Cryptography : Algorithms and Implementations Using C++’’***, Apple Academic Press Inc , Oakville, Canada, 31 Oct 2015.

[26] T. S. Denis, ***“Cryptography for Developers, Syngress Media’’***,Rockland, MA, United States, 01 Jan 2007.

[27] Web site for Springer International Publishing AG. Part of Springer Nature, ***“https://link.springer.com/referenceworkentry/10.1007%2F978-1-4419-5906-5_374’’***,Visited in 13 May 2018

[28] Web Site for Security Information Search, ***“https://searchsecurity.techtarget.com/definition/block-cipher’’***,visited in 13 May 2018

[29] J.-Y. Chouinard, ***“Design of Secure Computer Systems’’***, Notes on the Data Encryption Standard (DES), University of Laval ,Québec, Canada, September 23, 2002.

[30] H. Dobbertin, V. Rijmen , A. Sowa, ***“Advanced Encryption Standard - AES’’***, Berlin, Germany:, 15 Sep 2005.

[31] S. Murphy, C. Cid , Matthew Robshaw, ***"Algebraic Aspects of the Advanced Encryption Standard''***, New York, NY, United States: Springer-Verlag New York Inc., 30 Aug 2006.

[32] J. Katz, Y. Lindell, ***"Introduction to Modern Cryptography, Second Edition''***, Bosa Roca, United States: Taylor & Francis Inc, 18 Dec 2014.

[33] W. Stallings, ***"Cryptography and Network Security: Principles and Practice''***, Global Edition, Harlow, United Kingdom: Pearson Education Limited, 11 Oct 2016.

[34] V. Rijmen, J. Daemen, ***"The Design of Rijndael : AES - The Advanced Encryption Standard''***, Berlin, Germany: Springer-Verlag Berlin and Heidelberg GmbH & Co. KG, 12 Dec 2011.

[35] Web site for asking questions , ***"https://www.quora.com/What-make-Java-the-most-popular-programmin-language''***, visited in 28 May 2018.

[36] Official netbeans Web Site, ***"https://netbeans.org/features/index.html''***, Visited in 23 May 2018

[37] Official Wireshark Web Site ***"https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html''***,Visited in 24 May 2018.