

Sudan University of Science and Technology
College of Engineering
School of Electronics Engineering



Implementation of Distributive Authentication System Using a Security Token.

A Research Submitted in Partial fulfillment for the Requirements of
the Degree of B.Sc. (Honors) in Electronics Engineering

Prepared By:

- 1- Hishameldeen Sharafeldeen Wd Easa Zeyada.
- 2- Mohammed Abdelwahid Ahmed Elmaleeh.
- 3- Mohammed Osman Ahmed Hassan.
- 4- Rabiee Alfatih Rabiee Mohammed.

Supervised By:

Dr. Rania Abdelhameed Mukhtar

September, 2015

DECLARATION

*“An Investigation in Knowledge Pays the
Best Interest”*

DEDICATION

We Dedicate This Work to Our Families and
Many Friends Who Have Supported us Through-
out the Process.

ACKNOWLEDGEMENT

We would like to express our gratitude to our supervisor **Dr. Rania Abdelhameed Mukhtar**

For the continuous support, for her patience, and her motivation. Her guidance helped us in all the time of research and writing of this thesis.

We would like also to take this opportunity to express our gratitude to many people who have been instrumental and gave us help, support and well wishes.

ABSTRACT

Today, single factor authentication, e.g. passwords, is no longer considered secure in the internet and banking world. Easy-to-guess passwords, such as names and age, are easily discovered by automated password-collecting programs, In addition ,it's exhausted and slow, especially on a multiple devices (distributed) environment.

Two factor authentications have recently been introduced to meet the demand of organizations for providing stronger authentication options to its users. Using two identification factors require often an additional device.

In this project, a security token will be used, the token will gain access for multiple devices by writing a single password, the Bluetooth is the method selected for connecting the token and devices.

The proposed method guarantees that authenticating to services, is done in a very fast and a secure manner. In this model we develop a token software for laptop; design a Bluetooth communication model between the token (laptop) and devices that will be authenticated; complete the development of this token distributed authentication prototype system finally.

The proposed method has been implemented and tested. Initial results show the success of the proposed method.

المستخلص

()

في التطبيقات التي تتطلب درجة عالية من الأمان (المصارف و البنوك كمثال) , لعدة أسباب أبرزها لجوء المستخدمون لى كلمات سر ضعيفة قابلة للتخمين عن طريق برامج التخمين الآلي التي يستخدمها المخترقون و لصوص المعلومات.

نها تعتبر طريقة مرهقة و بطيئة خصوصاً

الكثير من الأجهزة التي قد تحولك بدورها الى أجهزة أخرى, فتصبح مسألة التحرك من جهاز خر مرهقة جسدياً و ذهنيّاً .

لهذه الأسباب تم استخدام الطرق ذات العاملين, حيث يكون العامل الأول هو كلمة السر يضاف عليه عامل آخر محسوس فيكون بمثابة مفتاح. عندها لا ينفع تخمين كلمة السر بدون وجود المفتاح الملموس و العكس صحيح, و بذلك تزيد درجة الأمان بطريقة هائلة.

هذا المشروع, و من خلال هذا البحث تم استخدام طريقة التحقق ذات العاملين حيث يمثل الأولى و هي

مكانية تخمين كلمة السر .

أما بالنسبة لمشكلة الارهاق الجسدي و الذهني و البطء المصاحب لعملية التحقق, فقد تم استخدام تقنية البلوتوث لربط الأجهزة المتعددة مع ذلك المفتاح المادي (الحاسوب) , فعندها سيتم المستخدم و كلمة السر مرة واحدة , ثم يتم التحقق في ثوان معدودة بعد أن ترسل (عبر البلوتوث) البيانات الخاصة بكل جهاز .

و لمزيد من الأمان, كان لابد من تشفير البيانات المرسله عبر البلوتوث, فتم ضافة خوارزمية التشفير.

تم تطبيق كل ما ذكر , حيث تم ربط المخدمات مع المفتاح

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	DECLARATION	I
	DEDICATION	II
	ACKNOWLEDGEMENT	III
	ABSTRACT	IV
	ABSTRACT IN ARABIC	V
	TABLE OF CONTENTS	VI
	LIST OF TABLES	IX
	LIST OF FIGURES	X
	LIST OF ABBREVIATIONS	XI
1	INTRODUCTION	
	1.1 preface	2
	1.2 Problem Statement	3
	1.3 Proposed Solution	3
	1.4 Research Aim And Objectives	4
	1.5 Methodology	4
	1.6 Thesis Outlines	5

2	LITERATURE REVIEW	
	2.1 Overview	7
	2.1.1 Distributive Authentication	7
	2.1.2 Rivest, Shamir and Adleman (RSA)	8
	2.1.3 Bluetooth	9
	2.2 Related Work Covered	10
3	Distributive Authentication System	
	3.1 Overview Of Distributive Authentication System	14
	3.2 System Description	14
	3.3 Bluetooth Communication Module	15
	3.3.1 Global Architecture	16
	3.3.2 Connection Establishment at Server Side	17
	3.3.3 Connection Establishment at Client Side	17
	3.4 Software Implementation	18
	3.4.1 Operating System	18
	3.4.2 Java	19
	3.4.3 Java Standard Edition	20
	3.4.4 Eclipse	20
	3.4.5 Structured Query Language	20
	3.4.6 Java Database Connectivity Technology	21
	(JDBC)	
	3.4.7 Terminal Session Connect (TSCON)	22
	3.4.8 Terminal Session Disconnect (TSDISCON)	22

	3.4.9 Code Explanation	23
	3.5 Hardware Implementation	25
	3.6 Model Setup	25
	3.6.1 Prototype Developed	26
	3.7 Expected Results	27
4	Results And Discussion	
	4.1 Results	31
	4.1.1 Security Token	31
	4.1.2 Server	33
5	CONCLUSION AND RECOMMENDATIONS	
	5.1 Conclusion	37
	5.2 Recommendations	37
6	REFERENCES	39
7	APPENDICES	42-63

LIST OF TABLES

TABLE NO	TITLE	PAGE
3.1	Important Methods and Functions of the Server.	23
3.2	Important Methods and Functions of the Security To- ken.	24

LIST OF FIGURES

FIGURE NO	TITLE	Page
2.1	Rivest, Shamir And Adelman.	9
3.1	System Description.	15
3.2	Client-Server Communication Using Bluetooth.	16
3.3	MYSQL Database.	21
3.4	TSCON Syntax And Parameters.	22
3.5	Bluetooth Circuit.	25
3.6	Model Setup.	26
3.7	How The Security Token Operate.	28
3.8	How The Server Interact With The Token.	29
4.1	Searching For Devices.	31
4.2	Searching For Services.	32
4.3	Adding Another Device.	32
4.4	Waiting For Incoming Connections.	33
4.5	Servers Profile Before “Login In”.	34
4.6	Servers Profile After “Login In”.	34
4.7	Servers Profile After “Log Out”.	35

LIST OF ABBREVIATIONS

AS	Authentication Server.
ATM	Auto Teller Machine.
BTSP	Bluetooth Serial Port Protocol.
FTP	File Transfer Protocol.
	Global System For Mobile Communication, Originally Group
GSM	Special Mobile.
GUI	Graphical User Interfaces.
HTTP	Hypertext Transfer Protocol.
ID	Identification.
IDE	Integrated Development Environment.
IMAP	Internet Message Access Protocol.
I/O	Input/Output.
JDBC	Java Database Connectivity.
JVM	Java Virtual Machine.
LAN	Local Area Network.
OS	Operating System.
OTP	One Time Password.
PC	Personal Computer.
PIN	Personal Identification Number.
PW	Password.
RDMBS	Relational Database Management System.
RSA	Ron Rivest, Adi Shamir And Leonard Adleman.
SE	Standard Edition.

SMS	Short Message Service.
SMTP	Simple Mail Transfer Protocol.
SQL	Structured Query Language.
TSCON	Terminal Session Connect.
URL	Uniform Resource Locator.
USB	Universal Serial Bus.

CHAPTER ONE

INTRODUCTION

1.1 Preface

A common problem is verifying that a given artifact was produced by a certain person or was produced in a certain place or period of history.

In computer science, verifying a person's identity is often required to secure access to confidential data or systems.

Authentication can come in many forms; Knowledge factors ("something only the user knows") are the most commonly used form of authentication. In this form, the user is required to prove knowledge of a secret in order to authenticate, another form is Possession factors ("something only the user has") which have been used for authentication for centuries, in the form of a key to a lock. A third form of authentication is the Inherence factors which are "something only the user is", or Biometric authentication. Biometric authentication also satisfies the regulatory definition of true multi-factor authentication [1].

Users may biometrically authenticate via their fingerprint, voiceprint, or iris scan using provided hardware and then enter a PIN or password in order to open the credential vault.

The usage of passwords for authentication is no longer sufficient and stronger authentication schemes are necessary. Strong authentication solutions require often two identification factors , in addition to the first factor "something you know" represented by passwords it is introduced a second factor "something you have" materialized by a security token [2].

1.1 Problem Statement

Most systems today rely on static passwords to verify the user's identity. For a system involving several devices, every device would have its own Password as basic requirement for distributed authentication. However, such passwords come with major management security concerns. Users tend to use easy-to-guess passwords, use the same password in multiple accounts, write the passwords or store them on their machines, Furthermore, hackers have the option of using many techniques to steal Passwords. Another issue is that if the user used a strong password but Forgot to log out from any device, this problem would leave the system Exposed to unauthorized users, and that is a security weakness.

1.2 Proposed Solution

In this research the following solution is proposed:

A security token will be used, the token will gain a distributive access for multiple devices with a specific password, Bluetooth will be the responsible for data communication, and the connection between the token and the servers will be encrypted.

1.3 Research Aim and Objectives

The main aim of this project is to design and implement distributed authentication system that relay on user associated token to provide an authentication mechanism to multiple devices environment e.g. data center. The detailed objectives of this project:

- To design and implement a prototype of the system that contains a token based distributed authentication and two servers.
- To implement security mechanism that increases the security at maintained usability level.
- To implement method for fast, easy and secure access, deny and lock the server devices via distributed authentication in windows based environment.

1.5 Methodology

In order to increase security, RSA algorithm had been used to encrypt data.

In order to provide a wireless access between devices, a wireless technology standard for exchanging data over short distances have been chosen, which is Bluetooth. Also Java programming languages (java standard edition) have been adopted as the main language to write codes of RSA algorithm and Bluetooth. Eclipse is the used software to compile and test these codes; Eclipse is an integrated development environment (IDE).it contains a base workspace and an extensible plug-in system for customizing the

environment. Bluecove (additional java libraries) has been added to eclipse, to work with Bluetooth classes.

1.6 Thesis Outlines

The thesis is divided into five chapters, Chapter one is an introduction that gives a background about the project, its aims and objectives, the problem statement and proposed solutions. It also gives a brief description on how to achieve those goals in the methodology.

Chapter two is a literature review that gives a brief review of authentication generally, and the timeline of using security tokens.

Chapter three is the system design (methodology) contains all the methods and steps in details in order to achieve the project objectives. Chapter Four presents the implementation procedure. Chapter five provides the conclusion and recommendations.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview

Authentication is one of the most important problems in security .it accrues whenever users have to provide credentials to prove their identity in order to access a computing resource. The goal of authentication is to ascertain that only legitimate users are granted access.

2.1.1 Distributive Authentication

The classic way by which people authenticate to computers (and by which computers authenticate to one another) is by supplying a Password. There are a number of problems with existing password Based schemes Which distributed authentication attempts to solve.

The goal of distributed authentication service is to provide authentication services in a distributed environment which are both more secure (more difficult for a bad guy to impersonate a good guy) and easier to use than existing mechanisms. In a distributed environment, authentication is particularly challenging. Users do not simply log on to one machine and use resources there. Users start processes on one machine which may request services on another. In some cases, the second system must request services from a third system on behalf of the user. Further, given current network technology, it is fairly easy to eavesdrop on conversations between computers and pick up any passwords that might be going by Distributed authentication service uses cryptographic mechanisms to provide strong, mutual authentication.

2.1.2 Rivest, Shamir and Adelman (RSA)

Data communication is an important aspect of our living. So, protection of data from misuse is essential. A cryptosystem defines a pair of data transformations called Encryption and decryption. Encryption is applied to the plain text i.e. the data to be communicated to produce cipher text i.e. encrypted data using encryption key.

Decryption uses the decryption key to convert cipher text to plain text i.e. the original data. Now, if the encryption key and the decryption key is the same or one can be derived from the other then it is said to be symmetric cryptography. This type of cryptosystem can be easily broken if the key used to encrypt or decrypt can be found. To improve the protection mechanism Public Key Cryptosystem was introduced in 1976 by Whitfield Diffe and Martin Hellman of Stanford University. It uses a pair of related keys one for Encryption and other for decryption. One key, which is called the private key, is kept secret and other one known as public key is disclosed.

The message is encrypted with public key and can only be decrypted by using the private key. So, the encrypted message cannot be decrypted by anyone who knows the public key and thus secure communication is possible. Rivest, Shamir and Adelman (RSA) introduced by Rivest, Shamir and Adelman as pictured in figure (2.1) is the most popular public key algorithm. It relies on the factorization problem of mathematics that indicates that given a very large number it is quite impossible in today's aspect to find two prime numbers whose product is the given number. As

we increase the number the possibility for factoring the number decreases [3][4].

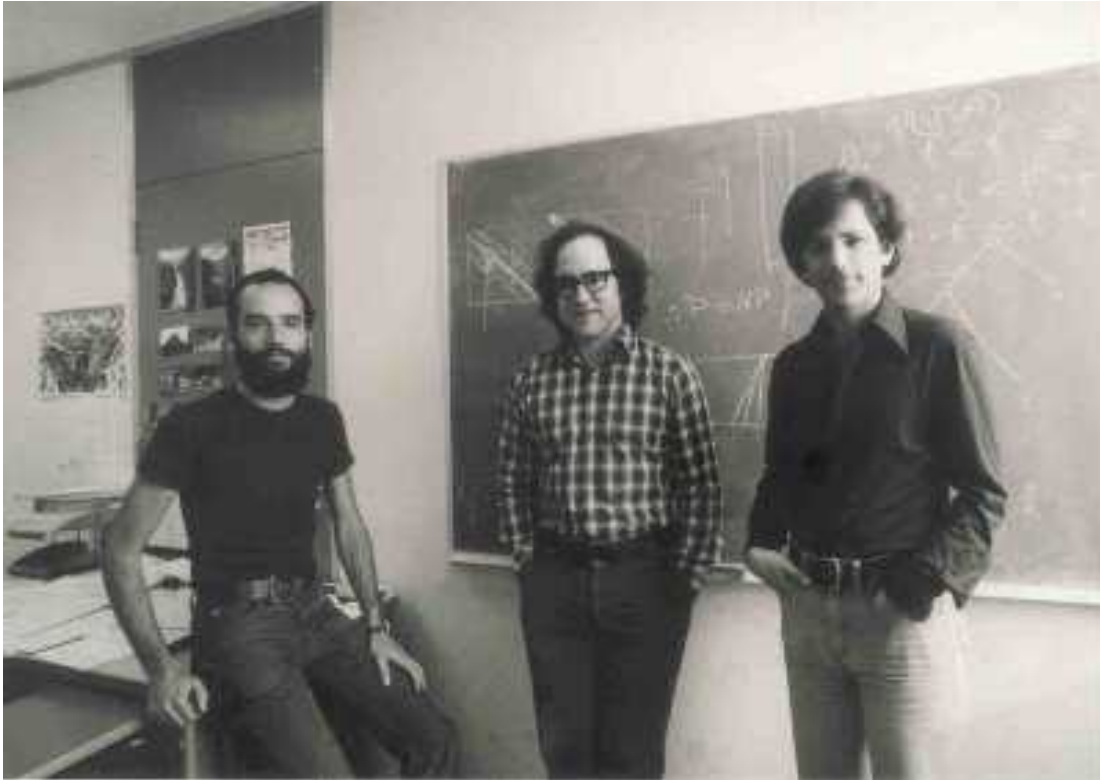


Figure 2.1: Rivest, Shamir and Adelman.

2.1.2 Bluetooth

Bluetooth is a wireless communication protocol which is use for short Distances and in low power consumption devices. Bluetooth can be used to Communicate to two are more other Bluetooth capable devices. Bluetooth is like any other communication protocol that we use every day like HTTP, FTP, SMTP, or IMAP. Bluetooth is also like these protocols in that it has client server architecture [5][6].

2.2 Related Work Covered

In [7], the proposed system is a mobile-based software token system that is supposed to replace existing hardware and computer-based software tokens. The proposed system is secure and consists of three parts:

- (1) Software installed on the client's mobile phone.
- (2) Server software.
- (3) A GSM modem connected to the server.

The system will have two modes of operation:

- **Connection-Less Authentication System:** A onetime password (OTP) is generated without connecting the client to the server. The mobile phone will act as a token and use certain factors unique to it among other factors to generate a one-time password locally. The server will have all the required factors including the ones unique to each mobile phone in order to generate the same password at the server side and compare it to the password submitted by the client. The client may submit the password online or through a device such as an ATM machine. A program will be installed on

The client's mobile phone to generate the OTP.

- **SMS-Based Authentication System:** In case the first method fails to work, the password is rejected, or the client and server are out of sync, the mobile phone can request the one time password directly from the server without the need to generate the OTP locally on the mobile phone. In order for the server to verify the identity of the user, the mobile phone sends to the server, via an SMS message, information unique to the user. The server

checks the SMS content and if correct, returns a randomly generated OTP to the mobile phone. The user will then have a given amount of time to use the OTP before it expires. Note that this method will require both the client and server to pay for the telecommunication charges of sending the SMS Message.

The concept of this proposal is described .a user wants to use some Web service on a mobile terminal, such as tablet or note PC, which do not hold a SIM card inside. It is supposed that such terminals have a wireless local area network (LAN) connection to the Internet. Usually, the service provider authenticates the user by using ID/PW pair. Instead of using the ID/PW pair, in our proposal, mobile phone is used as key device for user authentication. So, the system requires a mobile phone by the user. Under this concept, the SIM-based authentication can be used for any kind of terminals. However, there are some problems. First, the mobile terminal and mobile phone have to contact each other by some method. Second, service providers cannot use SIM-based authentication directly [8].

In[9], on this model of authentication, a user uses his mobile phone which works as an authentication token, that provide the authentication for laptop over a short-range wireless link. The user authenticate to the token infrequently. In tum, the mobile phone continuously authenticates to the laptop by means of the short-range, wireless link. The system design consists of three parts: laptop-cell phone authentication system, user authentication system and communication module.

On this system, implementation of zero-interaction authentication consists of two parts: an in-kernel encryption module and a user-level authentication

system. The kernel portion provides cryptographic I/O, manages file keys, and polls for the token's presence [10].

The authentication system consists of a client on the user's lap-top and a server on the token, communicating via a secured channel.

In [11] a Challenge Token-based Authentication as a second authentication factor is presented on this paper. This technique is based on two authentication factors, which is in addition to the first factor "user name and password", it also uses the client soft token that will be stored in a mobile phone or USB. The soft token will be obtained during registration and will never be transmitted during the authentication process. This token will be used by a mobile Client Program to generate a secure Authentication

Server (AS) public key in order to respond to the AS's challenge. Finally the strength of RSA algorithm is discussed in [12].

CHAPTER THREE

METHODOLOGY

3.1 Overview of Distributive Authentication System

The main aim of this thesis work is to provide an implementation of distributive authentication system using a security token at maintained usability level. Everything starting from communication module, software, hardware implementation to the final prototype will be explained on this chapter.

3.2 System Description

In order to increase security, RSA algorithm had been used to encrypt data.

In order to provide a wireless access between devices, a wireless technology standard for exchanging data over short distances have been chosen, which is Bluetooth. Also Java programming languages (java standard edition) have been adopted as the main language to write codes of RSA algorithm and Bluetooth. Eclipse is the used software to compile and test these codes; Eclipse is an integrated development environment (IDE).it contains a base workspace and an extensible plug-in system for customizing the environment. Bluecove (additional java libraries) has been added to eclipse, to work with Bluetooth classes. Figure 3.1 below describes the system.

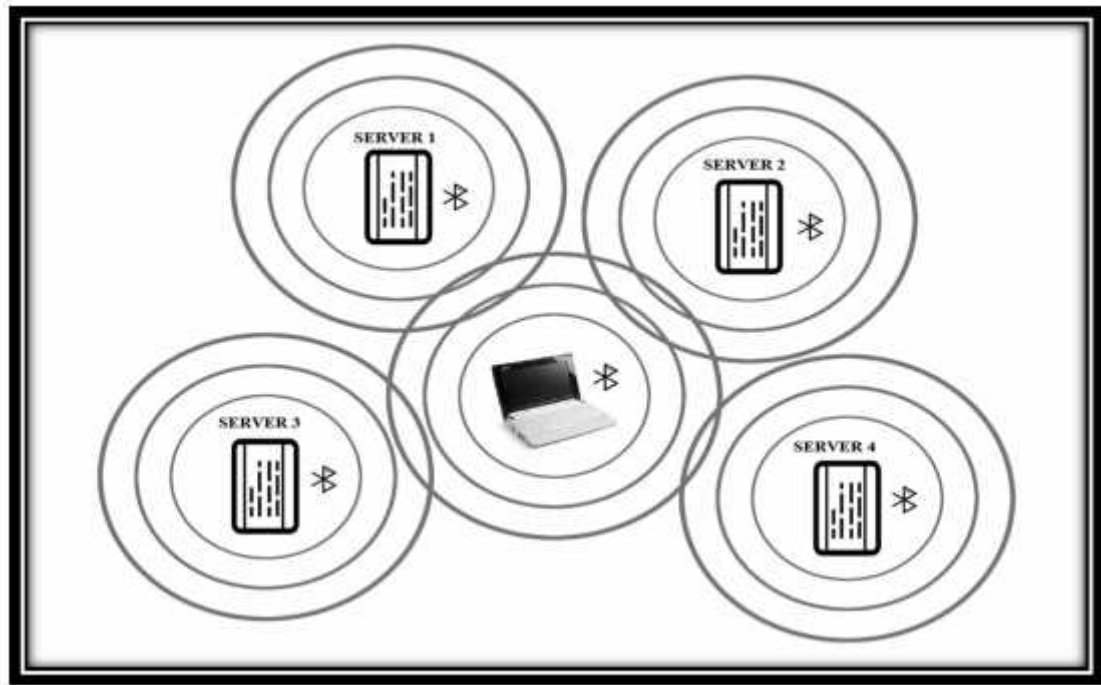


Figure 3.1: System Description.

3.3 Bluetooth Communication Module

Bluetooth is a cable-replacement technology or a means of wireless communication using radio transmission. It scores over infra-red transmission since it does not require line-of-sight contact and permits one-to-many communication. It is a standard for a small, cheap radio chip to be plugged into computers, printers, mobile phones, etc. Bluetooth chip transmits information at a special frequency to a receiver Bluetooth chip, which will then give the information received to the computer, phone whatever. In short, the owner of a Bluetooth-equipped device can initiate a search for all Bluetooth-equipped devices within the range of the device.

He/She can then request connection to the services of all or some of the found devices. Thus, a local network, called a Personal Area Network gets established and the devices in the network can freely exchange messages and data.

3.3.1 Global Architecture

When using Bluetooth technology, a prototype for supporting a service-oriented communication between a server and a client should be developed. ‘The server provides a list of services that a client can access interactively. The client retrieves the services desired through a process of communication via messages. The Figure 3.2 shows the global architecture’ [13].

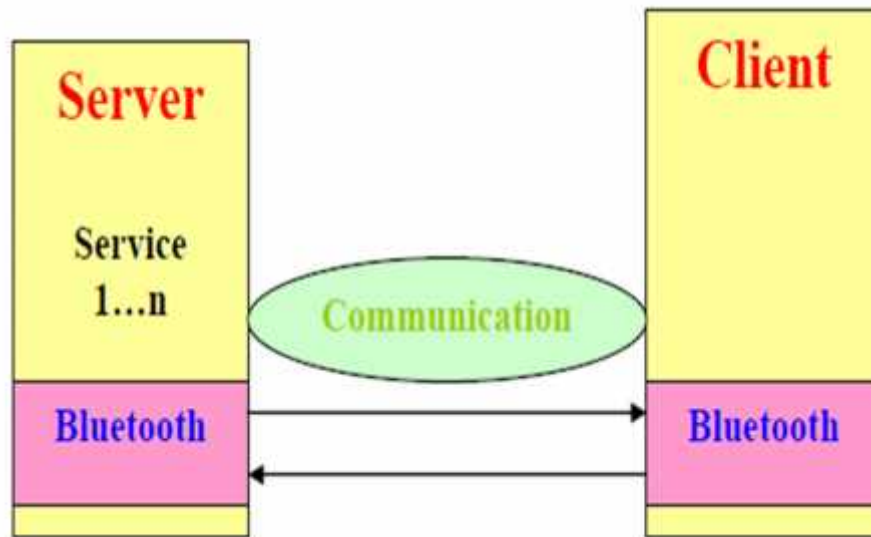


Figure 3.2: Client-Server Communication using Bluetooth.

Technically speaking, Bluetooth devices transmit at a frequency of 2.4 GHz and have a maximum range of up to 100m. The data transfer rate on a

Bluetooth connection can go up to 700 kbps, but depends upon the device and environmental factors. It can support both data and voice communications.

3.3.2 Connection Establishment at Server Side

For a server and client to communicate using the Serial Port Profile, each must perform a few simple steps. By the server, it must:

1. Construct a URL (Uniform Resource Locator) that indicates how to connect to the service, and store it in the service record.
2. Make the service record available to the client.
3. Accept a connection from the client.
4. Send and receive data to and from the client.

The URL may look something like:

btsp://[Bluetooth device address]:[channel number], where “btsp” stands for the Bluetooth Serial Port Protocol, Bluetooth device address is the address of the Bluetooth USB adapter and channel number is a number assigned dynamically by the server program.

3.3.3 Connection Establishment at Client Side

To set up an RFCOMM connection to a server the client must:

1. Construct a connection URL using the service record.
2. Initiate a service discovery to retrieve the service record.
3. Open a connection to the server.
4. Send and receive data to and from the server.

3.4 Software Implementation

In this part Operating system, important techniques, tools will be explained.

3.4.1 Operating System

‘Microsoft windows (or simply windows) is a family of graphical operating system developed, marketed and sold by Microsoft

Windows consist of several families of operating systems each of which cater to certain sector of the computing industry. Microsoft introduced an operating environment named Windows on November 20, 1985 as a graphical operating system shell for MS-DOS in response to the growing interest in graphical user interfaces. Microsoft Windows came to dominate the world's personal computer market with over 90% market share, overtaking Mac OS, which had been introduced in 1984’ [14]. There are several advantages of using windows:

- **Ease of use:** Users are familiar with first versions of Windows will probably also find the more modern ones easy to work with. This is ascribable to everything from the standardized look and feel of almost all programs written for Windows to the way the file system has been presented ever since the time of MS-DOS .this is one of the main reasons why Windows users are often refuse to switch operating systems.

- **Available software:** There is a huge selection of software available for Windows. This is both due to and the reason for Microsoft's dominance of the world market for PC computer operating systems and office software. If you're looking for an application to suit your business needs, chances are that if it exists there will be a Windows version of it available somewhere.
- **Support for new hardware:** Virtually all hardware manufacturers will offer support for a recent version of Windows when they go to market with a new product. Again, Microsoft's dominance of the software market makes Windows impossible for hardware manufacturers to ignore. So, if you run off to a store today any buy some random new piece of computer hardware, you'll find that it will probably work with the latest version of Windows.

3.4.2 Java

'Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA)' [15], meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture.

3.4.3 Java Standard Edition

Java platform, standard edition or java (SE) is widely used platform for development and deployment of portable applications for desktop and server environments. Java SE uses the object-oriented Java programming language. It is part of the Java software platform family.

3.4.4 Eclipse

Eclipse is an integrated development environment (IDE). 'It contains a base workspace and an extensible plug-in system for customizing the environment' [16]. To work with Bluetooth classes, an additional java libraries has been added to eclipse known as Bluecove.

3.4.5 Structured Query Language

'MySQL is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL)' [17]. SQL is the most popular language for adding, accessing and managing content in a database. It is most noted for its quick processing, proven reliability, ease and flexibility of use. It will used to store MAC addresses and a hashed password of each device as shown in figure 3.3.

```

C:\Program Files\MySQL\MySQL Server 5.0>mysql
Enter password: 
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.0.41-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use ht;
Database changed
mysql> select * from blue;
+----+-----+-----+
| name | PIC   | password |
+----+-----+-----+
| mhmt | 001091520230 | F4bf34f42adbf9552929a142f6c270fe |
| aze  | 001701000030 | 7f6e6000f0a7749eb6c406619254b9c |
| rahim | 201600000031 | 8102dabbb60b4adb980ebbb23f22fabf |
| hicham | 002713900741 | 439de6e2eeec1c8438d2714b887165a1 |
+----+-----+-----+
4 rows in set (0.00 sec)

mysql>

```

Figure 3.3: MYSQL Database.

3.4.6 Java Database Connectivity Technology (JDBC)

JDBC is a Java database connectivity technology (Java Standard Edition platform) this technology defines how a client may access a database. It provides methods for querying and updating data in a database. MySQL connector is an additional library has been added to the code to provide Java database connectivity.

3.4.7 Terminal Session Connect (TSCON)

TSCON command is used to connect to another Terminal Services user session. If there is more than one session on a server there, this option can be used to switch between the sessions. So in this case, TSCON command will be used to switch from one session (user account) to another session (administrator account). Figure 3.4 shows syntax and the parameters of TSCON command.



```
Administrator: Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>tscon

TSCON [sessionid | sessionname] [/DEST:sessionname]
      [/PASSWORD:pw | /PASSWORD:*] [/U]

sessionid      The ID of the session.
sessionname    The name of the session.
/DEST:sessionname Connect the session to destination sessionname.
/PASSWORD:pw   Password of user owning identified session.
/U            Displays information about the actions performed.
```

Figure 3.4: TSCON Syntax and Parameters.

3.4.8 Terminal Session Disconnect (TSDISCON)

The TSDISCON command can be used to disconnect an active Terminal Services session. The session remains attached to the Terminal Services server in a disconnected state. TSDISCON command will be used

to disconnect from one session (administrator account) back to another session (user account).

3.4.9 Code Explanation

In this section, important methods that has been used on java code will be clarified, each with a description of result of using it. The important functions of server will be explained in Table 3.1. Table 3.2 explains it on side of the client (security token).

Table 3.1: Important Methods and Functions of the Server.

Name	Description
SetDiscoverable	<ul style="list-style-type: none"> Makes server's device discoverable to the token.
areKeyspresent()	<ul style="list-style-type: none"> Checks /C: directory if it contains pair of keys, if it doesn't then "generateKey ()" Will be used
generateKey ()	<ul style="list-style-type: none"> Generates Pair of keys and save it on /C: directory
OpenOutputStream	<ul style="list-style-type: none"> Opens the output stream in order to send data (public key) over the stream of Bluetooth.
SendPublicKey	<ul style="list-style-type: none"> Sends server public key to the token.
OpenInputStream	<ul style="list-style-type: none"> Standby to Receives data over the stream of Bluetooth.
Available ()	<ul style="list-style-type: none"> Checks if the device is ready to receive data over Bluetooth.
Revebyte ()	<ul style="list-style-type: none"> Receives stream of bytes password.
Decrypt (Byte array [], private key)	<ul style="list-style-type: none"> Decrypts the received byte array using the private key.
GoAdmin	<ul style="list-style-type: none"> Execute the TSCON command to the server.
Logout ()	<ul style="list-style-type: none"> Execute TSDISCON command to the server.

Table 3.2: Important Methods and Functions of the Security Token.

Name	Description
LocalDeviceGetLocalDevice	<ul style="list-style-type: none"> Retrieve Bluetooth's MAC address of the local device.
GetDiscoveryAgent	<ul style="list-style-type: none"> Searches for the available Bluetooth devices.
DeviceDiscoverd	<ul style="list-style-type: none"> Gives a list of discovered devices.
InquiryCompleted	<ul style="list-style-type: none"> Stops the discovery.
ServicesSearchComplete	<ul style="list-style-type: none"> Stops the search of services.
ServicesDiscoverd	<ul style="list-style-type: none"> Shows a list of services found.
RevePublic key	<ul style="list-style-type: none"> Receives public keys from the clients.
Scanner (System.In)	<ul style="list-style-type: none"> Receives password from the user of the token.
ConsoleReadPassword	<ul style="list-style-type: none"> Hides the received password.
Text.CompareTo	<ul style="list-style-type: none"> Compares the received password with the one on the database (Authentication).
GetBluetoothAddress	<ul style="list-style-type: none"> Copies Bluetooth's MAC address of server device from database.
Database	<ul style="list-style-type: none"> Retrieve passwords from database.
Encrypt (string, public key)	<ul style="list-style-type: none"> Encrypts data (string) using server's public key, the encrypted data type would be Array of bytes.
OpenOutputStream	<ul style="list-style-type: none"> Opens the output stream in order to send data over the stream of Bluetooth.
Sendbyte	<ul style="list-style-type: none"> Sends stream of bytes (password).

3.5 Hardware Implementation

The hardware part of the work comprises three devices; two pcs have been chosen to represent the clients, and the third as a server. Each of those three devices, each contains a Bluetooth circuit as shown in figure 3.7.

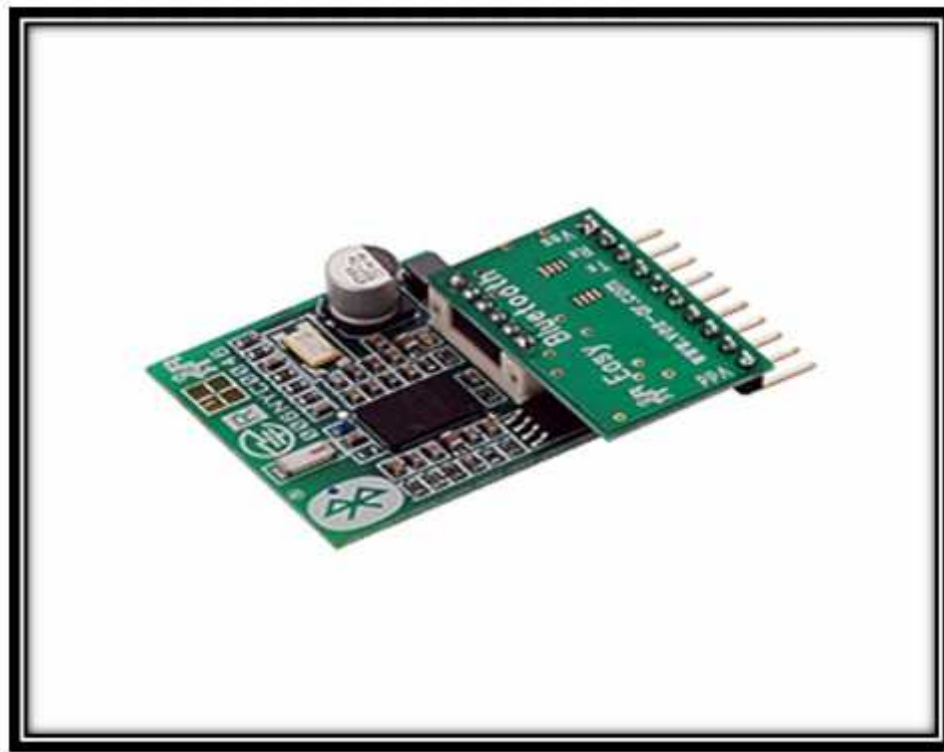


Figure 3.7 : Bluetooth Circuit.

3.6 Model Setup

As a prototype, three laptops have been used, two representing servers and one acting as a security token. See figure 3.8.



Figure 3.8: Model Setup.

3.6.1 Prototype Developed

The prototype developed includes two programs, one for the Server, and another for the token device. Server program runs on Windows 8.1 .It creates a new service called RFCOMM_Server that is based on Serial Port Profile. It registers this service in The Service Database, which normally stores the default services. It then opens An Input Stream connection and then blocks, waiting for a client to connect to it. Henceforth, the new service becomes detectable and connectable for all Bluetooth devices. On the token-side, the user starts client program also on Windows 8.1. Automatically detects the Server. The user then type “start” command on the console, in order to connect to the service.

If all goes well, the client program finds the service and connects to it. The user can now send password to the server by writing the global password in the text box displayed by client program, after that by typing “send”, each server’s password will be retrieved from the database then it will be distributed to the right device based on the MAC addresses saved on the database.

3.7 Expected Results

As an expected result, figure 3.5 shows how the security token should operate, and figure 3.6 shows how the server should interact with the token.

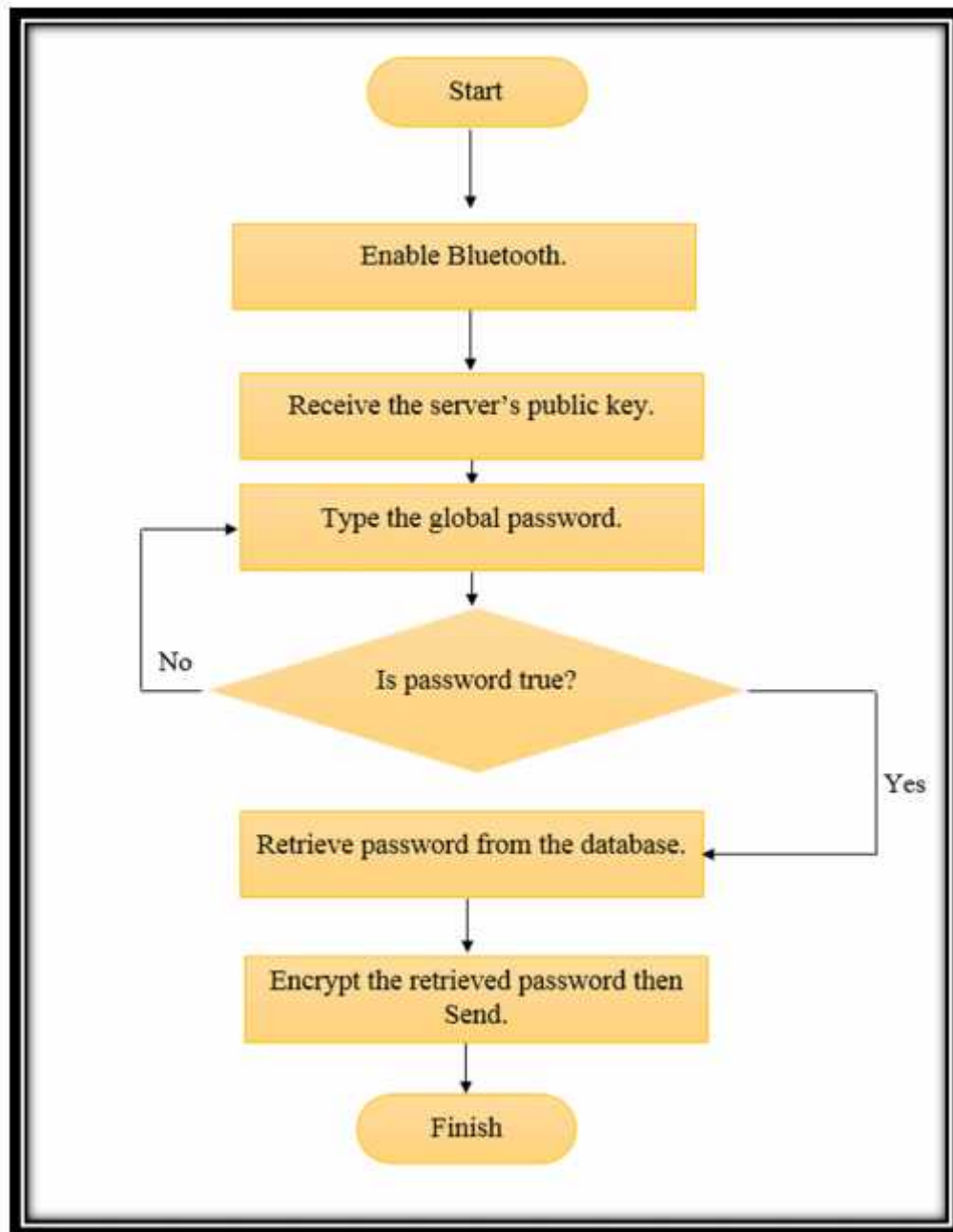


Figure 3.5: How the security token operates.

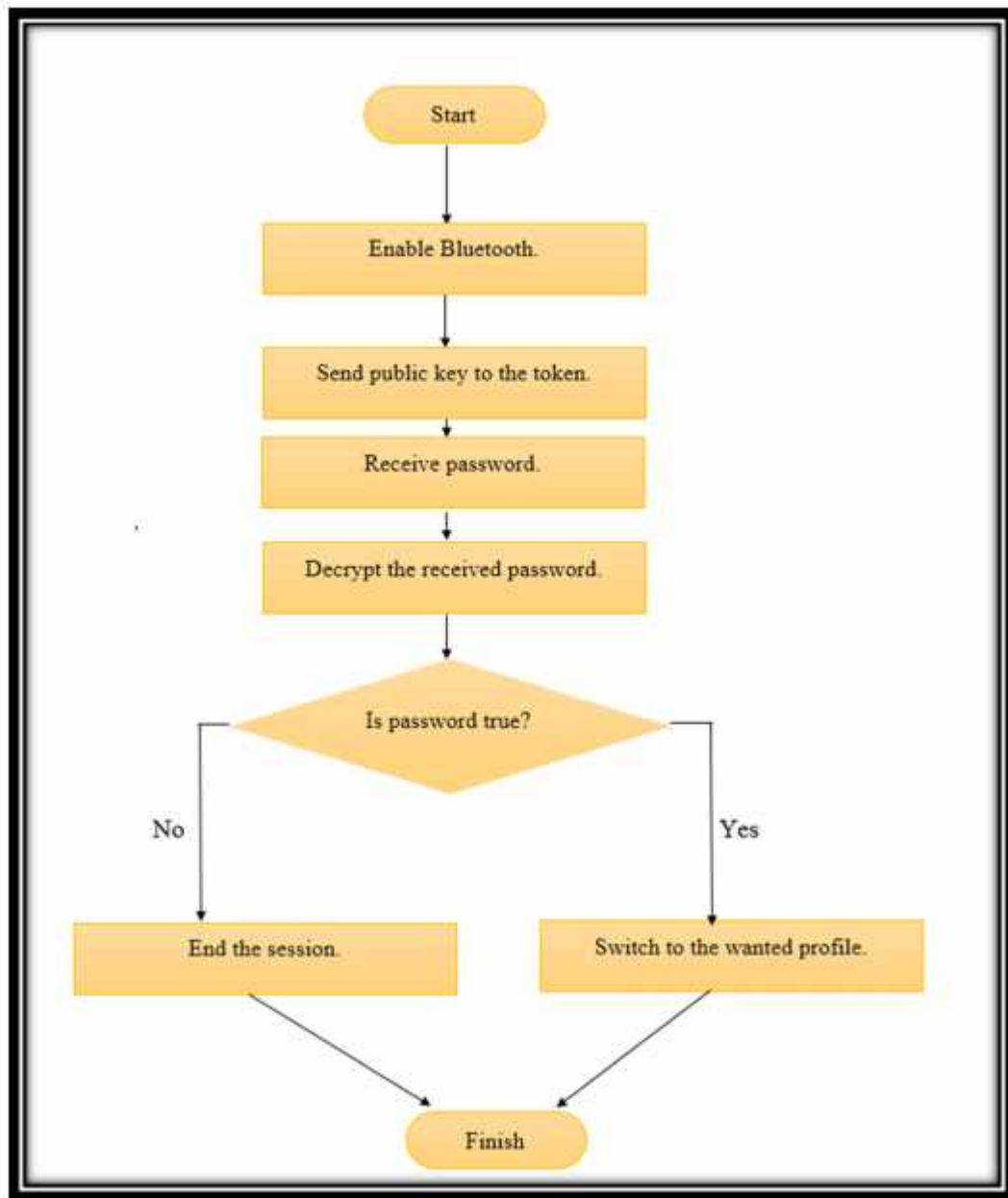


Figure 3.6: How the server interacts with the token.

CHAPTER FOUR

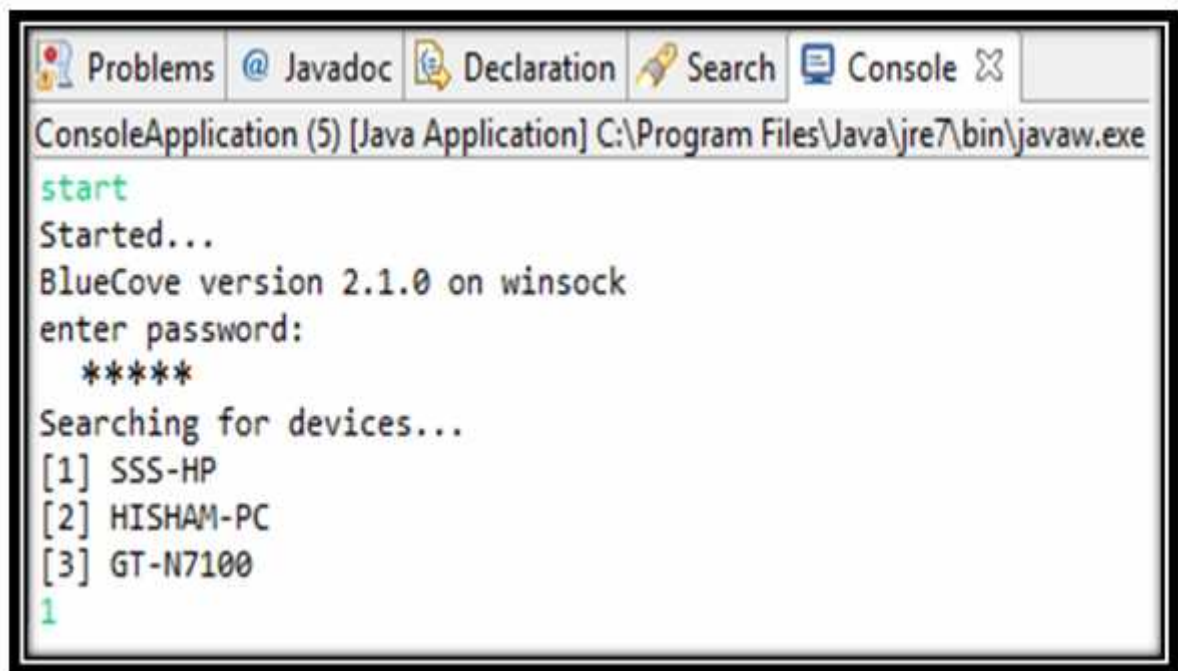
RESULTS AND DISCUSSION

4.1 Results

By compiling the code, the following output has occurred.

4.1.1 Security Token

By compiling the code on the security token, the device will start searching for devices, after that first device will be chosen as figure 4.1 shows.



```
start
Started...
BlueCove version 2.1.0 on winsock
enter password:
*****
Searching for devices...
[1] SSS-HP
[2] HISHAM-PC
[3] GT-N7100
1
```

Figure 4.1: Searching for Devices.

After choosing the wanted device, as a part of Bluetooth process, service number will be decided and the connection will be established as the figure below shows.

```
Searching for services...
[1] btspp://001F81000830:1;authenticate=true;encrypt=true;master=false
1
[Connecting to "btspp://001F81000830:1;authenticate=true;encrypt=true;master=false"...
Connection established... Streams opened...
```

Figure 4.2: Searching for Services.

Next, the user have to answer the question “Add another devices?” By typing “Yes”, the discovered devices will be listed again to choose from, figure 4.3 explains.

```
Add another Device?
Y
Searching for devices...
[1] SSS-HP
[2] HISHAM-PC
[3] GT-N7100
2
[2] btspp://001F81000830:1;authenticate=true;encrypt=true;master=false
```

Figure 4.3 Adding another Devices.

Finally, typing “login in” will switch the profile on the client’s device, and typing “logout” will sign out from the current profile.

4.1.2 Server

By compiling the code on the server side, the device will wait for incoming connections, then will execute the “login in” or “log out” commands after the stream is opened with the token. Figure 4.4 below shows a server on standby mode waiting for the connection.

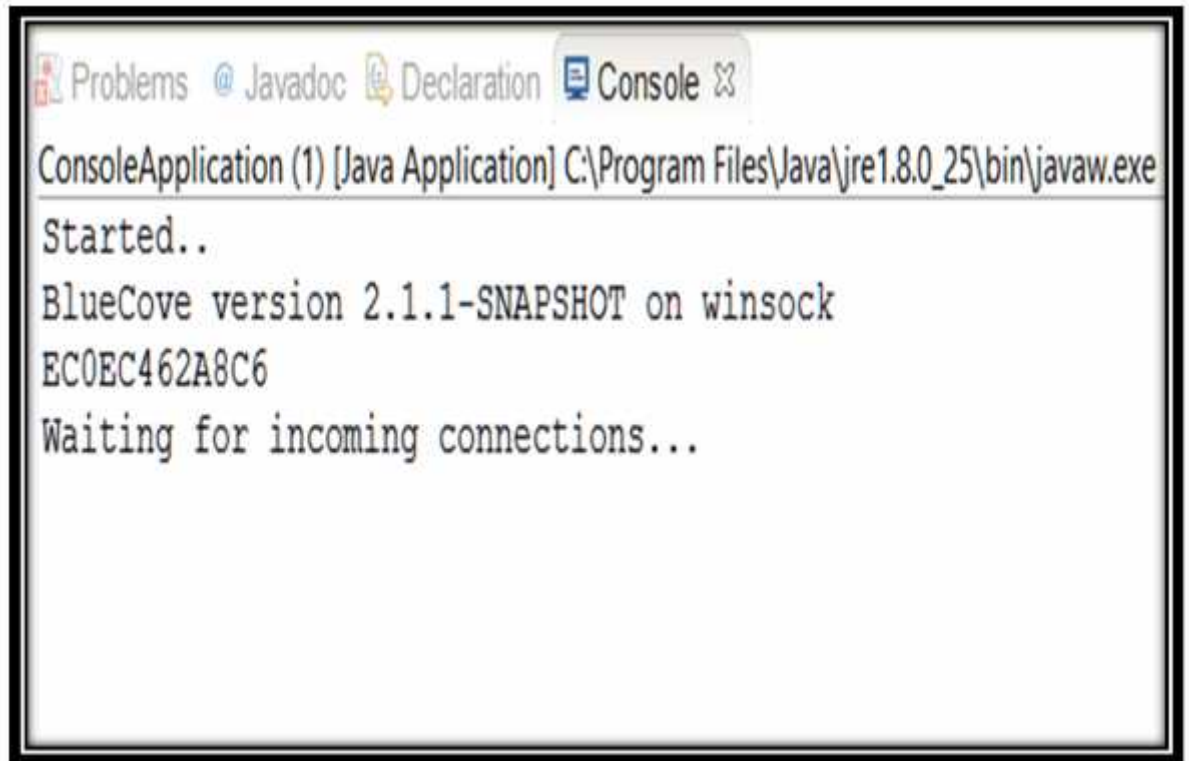


Figure 4.4: Waiting for Incoming Connections.

Figure 4.5 shows user profile before the execution of “login in” command, figure 4.6 shows administrator profile after the execution of “login in” command on the token. Figure 4.7 shows the result of executing “log out” command on the token.

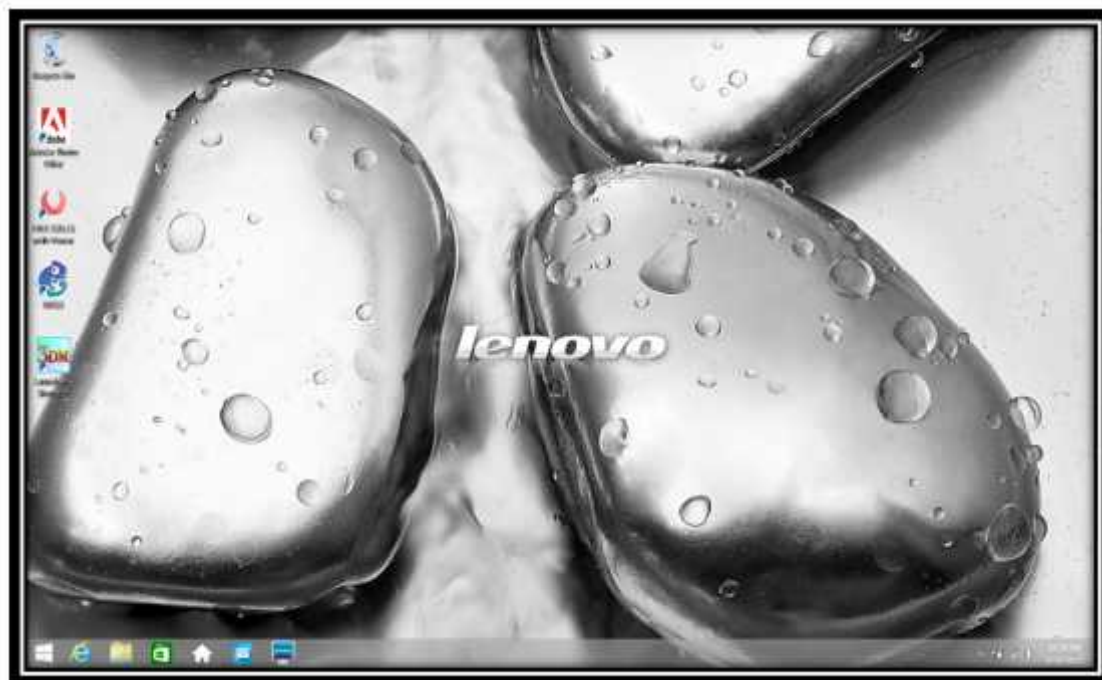


Figure 4.5: Servers Profile before “Login in”.



Figure 4.6: Servers Profile after “Login in”.

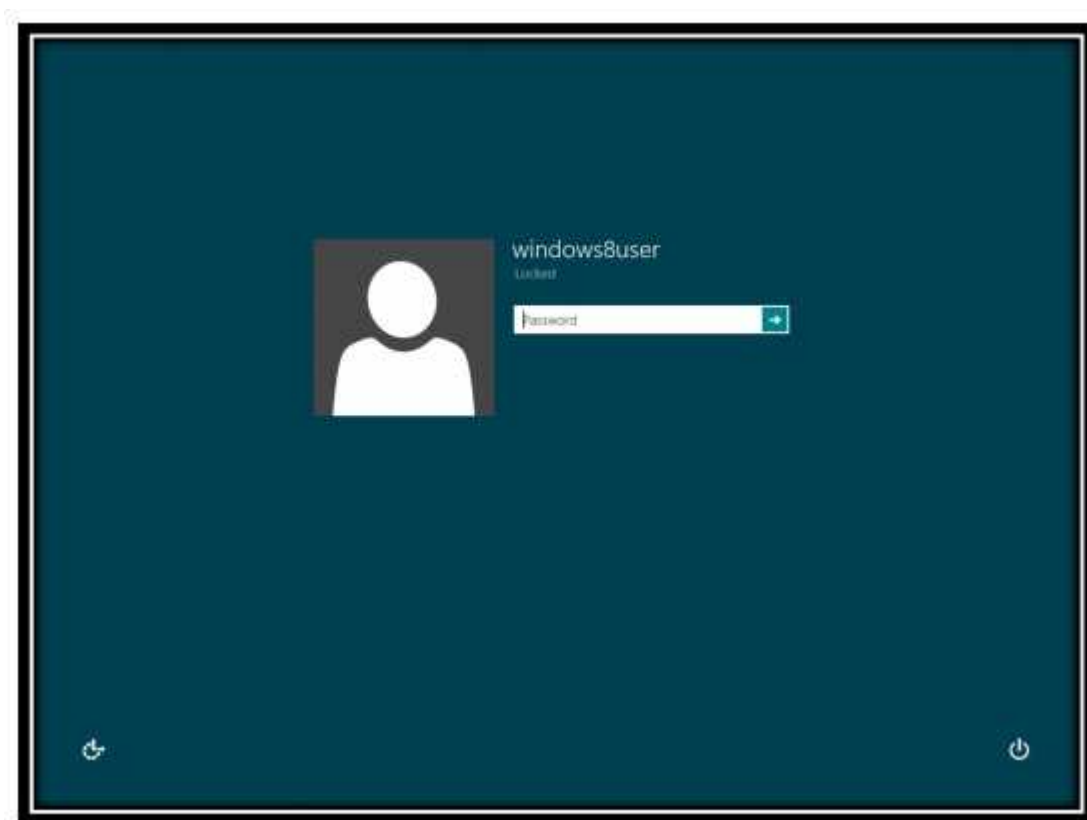


Figure 4.7: Servers Profile after “Log Out”.

CHAPTER FIVE

CONCLUSION

AND RECOMMENDATIONS

5.1 Conclusion

In order to implement method for fast, easy and secure access, deny and lock the server devices via distributed authentication, this project has suggested a token-based authentication scheme, which uses the user's personal computer as a token, Bluetooth wireless communication technology for data communication and RSA as a security algorithm. It can join the existing password authentication system, solve the security problems of static password authentication and can be used on multiple devices authentication on a distributed environment.

5.2 Recommendations and Future Work

In future work, the following points can be taken under consider:

- Use a mobile phone as a security token since it's commonly more associate with the user.
- Biometric authentication could be used instead of passwords.
- Remote Access could be configured on the security token.
- Machine to machine authentication could be a great application of distributive authentication, especially on supervisory control and data acquisition (SCADA) systems.

REFERENCES

- [1] B. Schneier, "Two-Factor Authentication: Too Little, Too Late," in Inside Risks 178, Communications of the ACM, 48(4), April 2005.
- [2] Mostarda, Leonardo; Dulay, Naranker; Dong, Changyu. "Place and Time Authentication of Cultural Assets". Retrieved 13 June 2014.
- [3] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler and J.D. Tygar, SPINS. "Security protocols for sensor networks, Mobile Computing and Networking", Rome, Italy, 2001.
- [4] David Pointcheval and Jacques Stern, "Security proofs for signature schemes, EUROCRYPT '96", Zaragoza, Spain, 1996.
- [5] David Meyer. "Bluetooth 3.0 released without ultra wideband". Retrieved 22 April 2009.
- [6] B. Hopkins and R. Antony, "Bluetooth for Java, First Edition", Apress, 2003
- [7] Fadi Aloul, Syed Zahidi "Two Factor Authentication Using Mobile Phones".2009.
- [8] Ryun Watanabe and Yutaka Miyake"User Authentication Method with Mobile Phone as Secure Token "2008.
- [9] Rania Abdelhameed, Sabira Khatun, Borhanuddin Mohd Ali and Abdul Rahman Ramli "Authentication model based Bluetooth enabled mobile phone" .2005.

- [10] Mark D. Corner, Brian D. Noble “Zero Interaction Authentication.” 2002.
- [11] Ghassan Kbar Associate Research Professor Riyadh Techno Valley, King Saud University Riyadh, Saudi Arabia. “Challenge Token-based Authentication – CTA”, 14/10/2011.
- [12] Elichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval and Jacques Stern, “RSAOAEP is secure under the RSA assumption, Journal of Cryptology”, 2002.
- [13] SHUYI LI “Mobile Business WS-04/05 Programming Bluetooth for mobile devices in Java”, 2012.
- [14] Microsoft Corporation “The Unusual History of Microsoft Windows”. Retrieved April 22, 2007.
- [15] Oracle. “ Design Goals of the Java Programming Language”. 1999-01-01. Retrieved 2013-01-.
- [16] Milinkovic, Mike. Innovation. “Building a Smarter Planet.” 3 November 2011.
- [17] Ohloh. Black Duck Software. “MySQL: Project Summary”. 17 September 2012.

APPENDICES

A. Client.java

```
import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.FileInputStream;

import java.io.ObjectInputStream;

import java.security.PrivateKey;

import java.security.PublicKey;

import javax.bluetooth.DiscoveryAgent;

import javax.bluetooth.LocalDevice;

import javax.microedition.io.Connector;

import javax.microedition.io.StreamConnectionNotifier;

public class Client extends ClientServerApplication

{

    StreamConnectionNotifier scn=null;


    public void go()

    {

        write("Started..\n");
```

```
try{ld=LocalDevice.getLocalDevice();} catch(Exception ex){ error(ex);
}

String address = ld.getBluetoothAddress();

System.out.println(address);


try{ld.setDiscoverable(DiscoveryAgent.GIAC);} catch(Exception ex){
error(ex); }


try{scn=(StreamConnectionNotifier)Connector.open("btspp://localhost:"+R
FCOMM_UUID+";name=rfcommtest;authorize=true");} catch(Exception
ex){ error(ex); }

write("Waiting for incoming connections...\n");

try{sc=scn.acceptAndOpen();} catch(Exception ex){ error(ex); }

write("Connection established... ");

try{is=new DataInputStream(sc.openInputStream());} catch(Exception
ex){System.out.println("hello");

error(ex); }

try{os=new DataOutputStream(sc.openOutputStream());}
catch(Exception ex){

System.out.println("hello");error(ex); }

write("Streams opened...\n");
```



```
    try {  
        // Check if the pair of keys are present else generate those.  
  
        if (!areKeysPresent()) {  
            // Method generates a pair of keys using the RSA algorithm and stores  
it  
            // in their respective files  
  
            generateKey();  
        }  
  
        //final String originalText = "Text to be encrypted ";  
  
        // Encrypt the string using the public key  
  
        ObjectInputStream  inputStream  =  new  ObjectInputStream(new  
        FileInputStream(PUBLIC_KEY_FILE));  
  
        final      PublicKey      publicKey      =      (PublicKey)  
inputStream.readObject();  
  
        sendPublickey(publicKey);  
  
  
        byte[] q = revebyte();
```

```
        inputStream      =      new      ObjectInputStream(new
FileInputStream(PRIVATE_KEY_FILE));

        final      PrivateKey      privateKey      =      (PrivateKey)
inputStream.readObject();

        final String plainText = decrypt(q, privateKey);

        System.out.println(plainText);

        GoAdmin(plainText);

        String l=recvText(is);

        logout();

        while(l=="kill"){

                System.out.println("killed");

        break;

        }

        try{os.close();} catch(Exception ex){ error(ex); }

        try{sc.close();} catch(Exception ex){ error(ex); }

        try{is.close();} catch(Exception ex){ error(ex); }

        inputStream.close();

        try{ld.setDiscoverable(DiscoveryAgent.NOT_DISCOVERABLE);}
        catch(Exception ex){ error(ex); }

        } catch (Exception e) {

                e.printStackTrace(); } } }
```

B. Server.java

```
import java.io.Console;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.security.PublicKey;

import java.sql.*;

import java.util.Scanner;

import java.util.Vector;

import javax.bluetooth.DeviceClass;

import javax.bluetooth.DiscoveryAgent;

import javax.bluetooth.DiscoveryListener;

import javax.bluetooth.LocalDevice;

import javax.bluetooth.RemoteDevice;

import javax.bluetooth.ServiceRecord;

import javax.bluetooth.UUID;

import javax.microedition.io.Connector;

import javax.microedition.io.StreamConnection;
```

```
public class Server extends ClientServerApplication implements
DiscoveryListener

{

    DiscoveryAgent da=null;

    Vector<RemoteDevice> dev=null;

    Vector<String> serv=null;

    static int c = 0;

    String sserv=null;

    RemoteDevice sdev=null;

    RemoteDevice sdev2=null;

    boolean running=false;

    boolean good;

    int key;

    boolean fs=false;

    String name;

    String MAC ;

    String password ;

    // JDBC driver name and database URL

    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";

    static final String DB_URL = "jdbc:mysql://localhost/bt";
```

```
// Database credentials

static final String USER = "mhmd";

static final String PASS = "123456789";

    private Scanner k;

    private Scanner input;

public void deviceDiscovered(RemoteDevice btDevice,DeviceClass cod)

{

    String name="";

    try{ name=btDevice.getFriendlyName(false)+" ";} catch(Exception ex){
error(ex); }

    dev.addElement(btDevice);

    write("[ "+dev.size()+" ] "+name+"\n");

}


public void inquiryCompleted(int discType)  {

}


    public void servicesDiscovered(int transID,ServiceRecord[]
servRecord){

}

    public void serviceSearchCompleted(int transID,int respCode){
```

```
    }  
  
    public String database(RemoteDevice sdev2) {  
  
        //STEP 1: Register JDBC driver  
  
        //STEP 2: Open a connection  
  
        //STEP 3: Execute a query to search for certain password based on  
the MAC  
  
        //STEP 4: Extract data from result set  
  
        //STEP 5: Clean-up environment  
  
    }  
}  
  
public void go() {  
    write("Started... \n");  
    try {  
        ld = LocalDevice.getLocalDevice();
```

```
    } catch (Exception ex) {  
        error(ex);  
    }
```

```
System.out.println("enter password:");
```

```
Console console =System.console();
```

```
String text = new String(console.readPassword());
```

```
String address = ld.getBluetoothAddress();
```

```
String deko=database(address);
```

```
while(!(text.compareTo(deko)==0))
```

```
{
```

```
    System.out.println("wrong password !!");
```

```
    return;
```

```
}
```

```
da = ld.getDiscoveryAgent();
```

```
dev = new Vector<RemoteDevice>();
```

```
while (dev.size() == 0) {  
    write("Searching for devices... \n");  
    running = true;  
    try {  
        da.startInquiry(DiscoveryAgent.GIAC, this);  
    } catch (Exception ex) {  
        error(ex);  
    }  
    while (running) {  
    }  
}  
good = false;  
while (!good) {  
    key = waitNumKey();  
    good = true;  
    try {  
        sdev = (RemoteDevice) dev.elementAt(key - 1);  
    } catch (Exception e) {  
        good = false;  
    }  
}
```



```
}

serv = new Vector<String>();

UUID[] uuidSet = new UUID[1];

uuidSet[0] = RFCOMM_UUID;

while (serv.size() == 0) {

    write("Searching for services... \n");

    running = true;

    try {

        da.searchServices(null, uuidSet, sdev, this);

    } catch (Exception ex) {

        error(ex);

    }

    while (running) {

    }

}

good = false;

while (!good) {

    key = waitNumKey();
```

```
        good = true;

        try {

            sserv = (String) serv.elementAt(key - 1);

        } catch (Exception e) {

            good = false;

        }

    }

String URL = sserv;

write("Connecting to \"" + URL + "\"...\n");

try {

    sc = (StreamConnection) Connector.open(URL);

} catch (Exception ex) {

    error(ex);

}

write("Connection established... ");

try {

    os = new DataOutputStream(sc.openOutputStream());

} catch (Exception ex) {

    error(ex);
```

```
}

try {

    is = new DataInputStream(sc.openInputStream());

} catch (Exception ex) {

    error(ex);

}

write("Streams opened...\n");

// conDev(sdev,is,os);

write("Add another Device?\n");

k = new Scanner(System.in);

String t = k.nextLine();

if (t.charAt(0) == 'y') {

    c=c+1;

    // conDev(sdev2,is1,os1);

    write("Started... \n");

    try {

        ld = LocalDevice.getLocalDevice();

    } catch (Exception ex) {

        error(ex);

    }

}
```

```
da = ld.getDiscoveryAgent();

dev = new Vector<RemoteDevice>();

while (dev.size() == 0) {

    write("Searching for devices... \n");

    running = true;

    try {

        da.startInquiry(DiscoveryAgent.GIAC, this);

    } catch (Exception ex) {

        error(ex);

    }

    while (running) {

    }

}

good = false;

while (!good) {

    key = waitNumKey();

    good = true;

    try {

        sdev2 = (RemoteDevice) dev.elementAt(key - 1);

    } catch (Exception e) {
```

```
        good = false;
    }
}

serv = new Vector<String>();
uuidSet[0] = RFCOMM_UUID;

while (serv.size() == 0) {
    write("Searching for services... \n");
    running = true;
    try {
        da.searchServices(null, uuidSet, sdev2, this);
    } catch (Exception ex) {
        error(ex);
    }
    while (running) {
    }
}

good = false;

while (!good) {
```

```
key = waitNumKey();

good = true;

try {

    sserv = (String) serv.elementAt(key - 1);

} catch (Exception e) {

    good = false;

}

}

String URL1 = sserv;

write("Connecting to \"" + URL1 + "\"...\n");

try {

    sc = (StreamConnection) Connector.open(URL1);

} catch (Exception ex) {

    error(ex);

}

write("Connection established... ");

try {

    os1 = new DataOutputStream(sc.openOutputStream());

} catch (Exception ex) {
```

```
        error(ex);
    }

    try {

        is1 = new DataInputStream(sc.openInputStream());

    } catch (Exception ex) {

        error(ex);

    }

    write("Streams opened...\n");

}

// System.out.println("checking the password...");

// System.out.println(" passwords loaded...");

password=database(sdev);

// System.out.println("distributive authentication sucessfully");

System.out.println("");

PublicKey publicKey1 = recevPublicKey(is);

byte[] cipherText = encrypt(password, publicKey1);

sendbyte(cipherText,os);
```

```
    if (c==0){  
        return;  
    }  
    else{  
        password=database(sdev2);  
  
        publicKey1 = recevPublicKey(is1);  
        cipherText = encrypt(password, publicKey1);  
  
        sendbyte(cipherText,os1);  
    }  
  
}
```

C. ClientServerApplication.java

```
import java.io.BufferedReader;  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.io.File;  
import java.io.FileNotFoundException;
```



```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectOutputStream;
import java.math.BigInteger;
import java.security.KeyFactory;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.KeySpec;
import java.security.spec.X509EncodedKeySpec;

import javax.bluetooth.LocalDevice;
import javax.bluetooth.UUID;
import javax.crypto.Cipher;
import javax.microedition.io.StreamConnection;

public abstract class ClientServerApplication extends ConsoleApplication
{
    static          final          UUID          RFCOMM_UUID=new
    UUID("6f4571008ba911dfa4ee0800200c9a66",false);
    LocalDevice ld=null;
    StreamConnection sc=null;
    DataInputStream is=null;
```

```
DataOutputStream os=null;
DataInputStream is1=null;
DataOutputStream os1=null;
DataInputStream is2=null;
DataOutputStream os2=null;

/**
 * String to hold name of the encryption algorithm.
 */
public static final String ALGORITHM = "RSA";

/**
 * String to hold the name of the private key file.
 */
public static final String PRIVATE_KEY_FILE = "C:/keys/private.key";

/**
 * String to hold name of the public key file.
 */
public static final String PUBLIC_KEY_FILE = "C:/keys/public.key";

public static void generateKey() {

}

/**
```

* The method checks if the pair of public and private key has been generated.

*

* @return flag indicating if the pair of keys were .generated.

*/

```
public static boolean areKeysPresent() { }
```

```
public static byte[] encrypt(String text, PublicKey key) {  
    // get an RSA cipher object and print the provider  
    // encrypt the plain text using the public key  
}
```

```
public static String decrypt(byte[] text, PrivateKey key) {  
    // get an RSA cipher object and print the provider  
    // decrypt the text using the private key  
}
```

```
boolean available(){ }
```

```
void sendText(String msg,DataOutputStream os) { }
```

```
public void GoAdmin(String ev) throws IOException {  
}
```

```
public static void logout() throws IOException {  
}
```

```
String recvText(DataInputStream is){
```

```
}
```

```
byte[] revebyte(){  
    }
```

```
PublicKey recevPublicKey(DataInputStream is)  
{  
    }
```

```
void sendbyte(byte[] r,DataOutputStream os)  
{  
    }
```

```
void sendPublickey(PublicKey number)  
{  
    }  
}
```