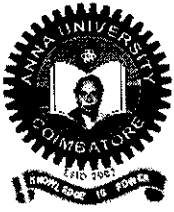


P-3618



**VIRTUAL ENERGY BASED ENCRYPTION AND
KEYING FOR WIRELESS SENSOR NETWORKS**



PROJECT REPORT

Submitted by

ARUN KUMAR.C

Reg.No: 0710108064

HARI JAYANTH.S

Reg.No: 0710108017

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of

Technology, Coimbatore)

COIMBATORE – 641 049

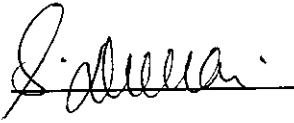
APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE-641 049

BONAFIDE CERTIFICATE

Certified that this project report entitled "VIRTUAL ENERGY BASED ENCRYPTION AND KEYING FOR WIRELESS SENSOR NETWORKS" is the bonafide work of C.ARUN KUMAR and S.HARI JAYANTH who carried out the research under my supervision. Certified also, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.



SIGNATURE

Mrs.P.Devaki.,M.E.,(Ph.D)

HEAD OF THE DEPARTMENT

Department of Computer

Science and Engineering

Kumaraguru College of Technology

Coimbatore-641049



SIGNATURE

Mr.M.Muthukumar.,M.E.,

ASSISTANT PROFESSOR

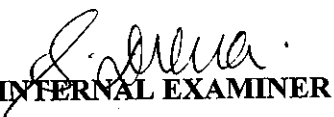
Department of Computer

Science and Engineering

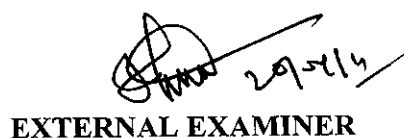
Kumaraguru College of Technology

Coimbatore-641049

The candidate with University Register Nos. 0710108004 and 0710108017 were examined by me in the project viva-voce examination held on 20-4-11



INTERNAL EXAMINER



EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project entitled "VIRTUAL ENERGY BASED ENCRYPTION AND KEYING FOR WIRELESS SENSOR NETWORKS" is a record of original work done by us and to the best of our knowledge, a similar work has not been submitted to Anna University or any Institutions, for fulfillment of the requirement of the course study.

The report is submitted in partial fulfillment of the requirement for the award of the Degree of Bachelor of Computer Science and Engineering of Anna University, Coimbatore.

Place: Coimbatore

Date: 20-4-11

Arun Kumar

(C.ARUN KUMAR)

Shari Jayanth

(S.HARI JAYANTH)

ACKNOWLEDGEMENT

We are intend to express our heartiest thanks to our chairman **Arutselvar** **Mr.N.Mahalingam.,B.sc.,F.I.E.,**and our co-chairman **Dr.B.K.Krishnaraj** **anavarayar, B.Com., B.L.,** and the correspondent **M.Balasubramaniam,** **I.com., M.B.A.,** for given us this opportunity to embark on this project.

We extend our sincere thanks to our Principal, **Dr. S.Ramachandran** **h.D.,** Kumaraguru College of Technology, Coimbatore, for being a constant source of inspiration and providing us with the necessary facility to work on this project.

We would like to make a special acknowledgement and thanks to **Dr. S. Thangasamy Ph.D.,** Dean of Research and Development, for his support and encouragement throughout the project.

We are intend to express our heartiest thanks to **Mrs.P.Devaki** **M.E,(Ph.D),** *Project coordinator* ,Head of the Department of Computer Science &Engineering, for her valuable guidance and useful suggestions during the course of this project.

Our deep gratitude and gratefulness to **our Guide Mr.M.Muthukumar.,M.E.,** **Assistant professor** Department of Computer Science & Engineering, for his supervision, enduring patience, active involvement and guidance.

We would like to convey our honest thanks to **all Faculty members** of the Department for their enthusiasm and wealth of experience from which we have greatly benefited.

We also thank our **friends and family** who helped us to complete this project fruitfully.

ABSTRACT

Designing cost-efficient, secure network protocols for Wireless Sensor Networks (WSNs) is a challenging duty because of resource-limited wireless devices. Since the communication cost is the most dominant factor in a sensor's energy consumption, an energy-efficient Virtual Energy-Based Encryption and Keying (VEBEK) scheme for WSNs is introduced that significantly reduces the number of transmissions needed for rekeying to avoid stale keys. In addition to the goal of saving energy, minimal transmission is imperative for some military applications of WSNs where an adversary could be monitoring the wireless spectrum.

VEBEK is a secured communication framework where sensed data is encoded using a scheme based on a permutation code generated via the RC4 encryption mechanism. The key to the RC4 encryption mechanism dynamically changes as a function of the residual virtual energy of the sensor. Thus, a one-time dynamic key is employed for one packet only and different keys are used for the successive packets of the stream. The intermediate nodes along the path to the sink are able to verify the authenticity and integrity of the incoming packets using a predicted value of the key generated by the sender's virtual energy, thus requiring no need for specific rekeying messages.

VEBEK is able to efficiently detect and filter false data injected into the network by malicious outsiders. The VEBEK framework consists of two operational modes (VEBEK-I and VEBEK-II), each of which is optimal for different scenarios. In VEBEK-I, each node monitors its one-hop neighbours where VEBEK-II statistically monitors downstream nodes.

In this project work, VEBEK is able to eliminate malicious data from the network incurring transmission overhead (increasing packet size or sending control messages for rekeying) in an energy efficient manner.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	
	LIST OF FIGURES	
	LIST OF TABLES	
	LIST OF ABBREVIATIONS	
1.	INTRODUCTION	1
	1.1 Project Overview	1
	1.2 Existing System	6
	1.3 Proposed System	7
2.	LITERATURE SURVEY	8
3.	REQUIREMENT SPECIFICATIONS	10
	3.1 Hardware Specifications	10
	3.2.1 Language Specification	10
	3.2.2 Introduction to .Net	10
	3.2.3 The .Net Framework	11
	3.2.4 Features of C# .Net	28
	3.2.5 Languages supported by .net	13
	3.3 Feasibility Study	15
	3.3.1 Economical Feasibility	15
	3.3.2 Technical Feasibility	16
	3.3.3 Social Feasibility	16
4.	SYSTEM DESIGN	
	4.1 Overall Architecture	17
	4.2 UML Representation	18
	4.2.1 Use Case Diagram	18
	4.2.2 Sequence Diagram	19
5.	SYSTEM DESCRIPTION	20
	5.1 Networking Module	20

	5.2 Stream Chirping Module	21
	5.3 RC4 Mechanism	22
	5.4 Authentication Module	23
	5.4 Key Recovery Module	24
6.	SYSTEM IMPLEMENTATION	25
	5.1 Networking Module	25
	5.2 Stream Chirping Module	29
	5.3 RC4 Mechanism	35
	5.4 Authentication Module	40
	5.4 Key Recovery Module	45
7.	SYSTEM TESTING	49
	7.1 Testing and Various Methodologies	49
	7.2 Test Cases	52
8.	FUTURE ENHANCEMENT	54
9.	CONCLUSION	55
10.	REFERENCES	56

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
1.1	RC4 Mechanism	4
4.1	Architecture Diagram	17
4.2	Use Case Diagram	18
4.3	Sequence Diagram	19
5.1	Sequence diagram for networking module	20
5.2	Sequence diagram for stream chirping module	21
5.3	Sequence diagram for RC4 mechanism	22
5.4	Sequence diagram for Key recovery module	23
5.5	Sequence diagram for Networking module	24
6.1	Snapshot of system main page	28
6.2	Snapshot of system after key generation	29
6.3	Snapshot for group name input	34
6.4	Snapshot for IP list and IP authentication	35
6.5	Snapshot for transfer operation	40
\ 6.6	Snapshot for summary and transfer operation	41
6.7	Snapshot for client IP verification	45
6.8	Snapshot for IP verification successful	46
6.9	Snapshot for iteration key number	48
6.10	Snapshot for key recovery	49

LIST OF TABLES

FIGURE NO	TITLE	PAGE NO
7.1	Case Generation Report	53

LIST OF ABBREVIATIONS

WSN	Wireless Sensor Networks
MAC	Message Authentication Codes
RC4	Rivest Code 4
IC	Integrated circuit
WEP	Wireless Encryption Protocol
SSL	Secure Socket Layer

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW:

Designing Cost-efficient and Secure Framework for Wireless Sensor Networks provides a technique to verify data in line and drop false packets from malicious nodes, thus maintaining the health of the sensor network. This system dynamically updates keys without exchanging messages for key renewals and embeds integrity into packets as opposed to enlarging the packet by appending Message Authentication Codes (MACs). Specifically, each sensed data is protected using a simple encoding scheme based on a permutation code generated with the RC4 encryption scheme and sent toward the sink.

A one-time dynamic key is used for one message generated by the source sensor and different keys are used for the successive packets of the Stream. The nodes those forwarding the data along the path to the sink are able to verify the authenticity and integrity of the data and to provide non-repudiation.

WIRELESS SENSOR NETWORKS

Sensors integrated into structures, machinery, and the environment, coupled with the efficient delivery of sensed information, could provide tremendous benefits to society. Potential benefits include: fewer catastrophic failures, conservation of natural resources, improved manufacturing productivity, improved emergency response, and enhanced homeland security . However, barriers to the widespread use of sensors in

structures and machines remain. Bundles of lead wires and fiber optic “tails” are subject to breakage and connector failures.

Long wire bundles represent a significant installation and long term maintenance cost, limiting the number of sensors that may be deployed, and therefore reducing the overall quality of the data reported. Wireless sensing networks can eliminate these costs, easing installation and eliminating connectors. The ideal wireless sensor is networked and scalable, consumes very little power, is smart and software programmable, capable of fast data acquisition, reliable and accurate over the long term, costs little to purchase and install, and requires no real maintenance. Selecting the optimum sensors and wireless communications link requires knowledge of the application and problem definition. Battery life, sensor update rates, and size are all major design considerations.

Examples of low data rate sensors include temperature, humidity, and peak strain captured passively. Examples of high data rate sensors include strain, acceleration, and vibration. Recent advances have resulted in the ability to integrate sensors, radio communications, and digital electronics into a single integrated circuit (IC) package. This capability is enabling networks of very low cost sensors that are able to communicate with each other using low power wireless data routing protocols. A wireless sensor network (WSN) generally consists of a base station (or “gateway”) that can communicate with a number of wireless sensors via a radio link. Data is collected at the wireless sensor node, compressed, and transmitted to the gateway directly or, if required, uses other wireless sensor nodes to forward data to the gateway. The transmitted data is then presented to the system by the gateway connection. The purpose of this CHAPTER is to provide a brief technical introduction to wireless

sensor networks and present a few applications in which wireless sensor networks are enabling.

DYNAMIC KEYING SCHEMES

Dynamic keying schemes go through the phase of rekeying either periodically or on demand as needed by the network to refresh the security of the system. With rekeying, the sensors dynamically exchange keys that are used for securing the communication. A one-time dynamic key is used for one message generated by the source sensor and different keys are used for the successive packets of the stream. The nodes forwarding the data along the path to the sink are able to verify the authenticity and integrity of the data and to provide no repudiation. The protocol is able to continue its operations under dire communication cases as it may be operating in a high-error-prone deployment area like under water.

Dynamic key management schemes perform keying functions (rekeying) either periodically or on demand as needed by the network. The sensors dynamically exchange keys to communicate. Although dynamic schemes are more attack resilient than static ones, one significant disadvantage is that they increase the communication overhead due to keys being refreshed or redistributed from time to time in the network. There are many reasons for key refreshment, including: updating keys after a key revocation has occurred, refreshing the key such that it does not become stale, or changing keys due to dynamic changes in the topology.

RC4 MECHANISM

Ronald Rivest of RSA developed the RC4 algorithm, which is a shared key stream cipher algorithm requiring a secure exchange of a shared key. The algorithm is

used identically for encryption and decryption as the data stream is simply XORed with the generated key sequence.

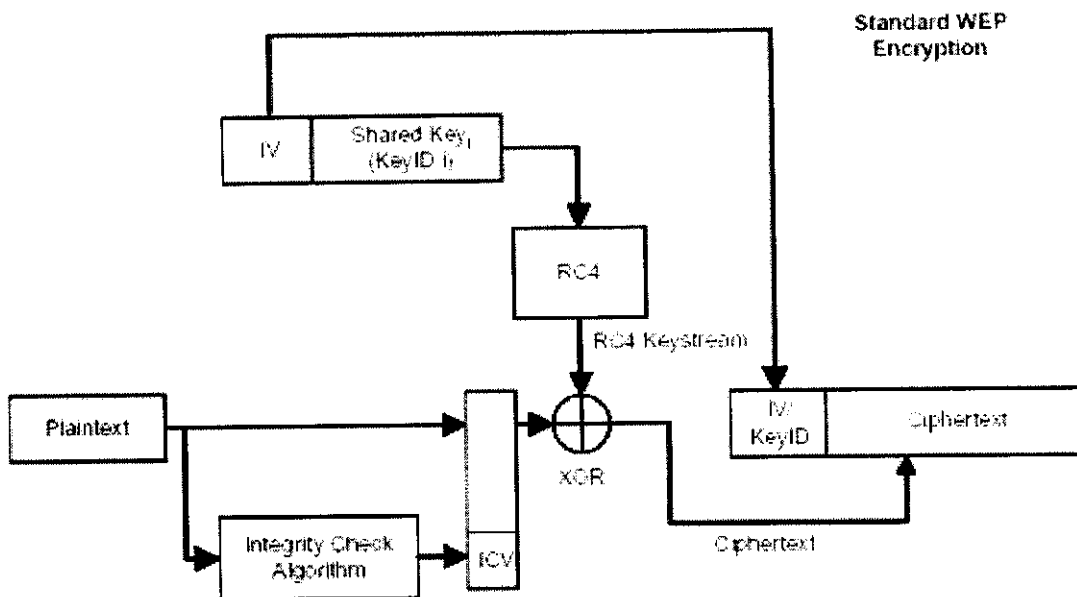


Fig 1.1 RC4 mechanism

The algorithm is serial as it requires successive exchanges of state entries based on the key sequence. Hence implementations can be very computationally intensive..This algorithm has been released to the public and is implemented by many programmers. This encryption algorithm is used by standards such as IEEE 802.11 within WEP (Wireless Encryption Protocol) using a 40 and 128-bit keys. Published procedures exist for cracking the security measures as implemented in WEP.

The VOCAL implementation of the RC4 algorithm is available in several forms. The forms include pure optimized software and varying levels of hardware complexity utilizing UDI instructions. The operations are supported using UDI

instructions for improved performance. When special assistance hardware is not available (as is the case on most general purpose processors), the byte manipulation/exchange operations are implemented via software.

In the algorithm the key stream is completely independent of the plaintext used. An $8 * 8$ S-Box (S0 S255), where each of the entries is a permutation of the numbers 0 to 255, and the permutation is a function of the variable length key. There are two counters i , and j , both initialized to 0 used in the algorithm.

ALGORITHM FEATURES

- Uses a variable length key from 1 to 256 bytes to initialize a 256-byte state table. The state table is used for subsequent generation of pseudo-random bytes and then to generate a pseudo-random stream which is XORed with the plaintext to give the ciphertext. Each element in the state table is swapped at least once.
- The key is often limited to 40 bits, because of export restrictions but it is sometimes used as a 128 bit key. It has the capability of using keys between 1 and 2048 bits. RC4 is used in many commercial software packages such as Lotus Notes and Oracle Secure SQL.
- The algorithm works in two phases, key setup and ciphering. Key setup is the first and most difficult phase of this encryption algorithm. During a N-bit key setup (N being your key length), the encryption key is used to generate an encrypting variable using two arrays, state and key, and N-number of mixing operations. These mixing operations consist of swapping bytes, modulo operations, and other formulas. A modulo operation is the process of yielding a remainder from division. For example, $11/4$ is 2 remainder 3; therefore eleven mod four would be equal to three.

- The algorithm works in two phases, key setup and ciphering. Key setup is the first and most difficult phase of this encryption algorithm. During a N-bit key setup (N being your key length), the encryption key is used to generate an encrypting variable using two arrays, state and key, and N-number of mixing operations. These mixing operations consist of swapping bytes, modulo operations, and other formulas. A modulo operation is the process of yielding a remainder from division. For example, $11/4$ is 2 remainder 3; therefore eleven mod four would be equal to three.

OBJECTIVE OF THE PROJECT

The objective of our system is to provide a cost-efficient and secure framework for Wireless Sensor Networks using efficient key management schemes. Here we use a secret key value in both server and client side to transfer data in a safe and efficient manner in wireless sensor networks. The secret key cannot be hacked by intruders because its designed in such a way to provide a secure path for communication.

1.2 EXISTING SYSTEM:

One significant aspect of confidentiality research in WSNs entails designing efficient key management schemes. This is because regardless of the encryption mechanism chosen for WSNs, the keys must be made available to the communicating nodes. The keys could be distributed to the sensors before the network deployment or they could be redistributed (re-keying) to nodes on demand as triggered by keying events. The former is static key management.

DISADVANTAGES

- They increase the communication overhead due to keys being refreshed or redistributed from time to time in the network.

- There are many reasons for key refreshment, including: updating keys after a key revocation has occurred, refreshing the key such that it does not become stale.

.3 PROPOSED SYSTEM

In addition to the goal of saving energy, minimal transmission is imperative for some military applications of WSNs where an adversary could be monitoring the wireless spectrum. Our system is a secure communication framework where sensed data is encoded using a scheme based on a permutation code generated via the RC4 encryption mechanism.

The key to the RC4 encryption mechanism dynamically changes as a function of the residual virtual energy of the sensor. Thus, a one-time dynamic key is employed for one packet only and different keys are used for the successive packets of the stream.

ADVANTAGES:

- Modular and flexible security architecture with simple technique for ensuring authenticity, integrity, and no repudiation of data without enlarging packets with MAC.
- Robust secure communication framework that is operational in direct communication situations and over unreliable medium access control layers.

CHAPTER 2

LITERATURE SURVEY

One significant aspect of confidentiality research in WSNs entails designing efficient key management schemes. This is because regardless of the encryption mechanism chosen for WSNs, the keys must be made available to the communicating nodes.

2.1 EFFICIENT KEYING MECHANISMS

We present a simple analysis for the rekeying cost with and without the transmission of explicit control messages. Rekeying with control messages is the approach of existing dynamic keying schemes whereas Rekeying without extra control messages is the primary feature here.

2.2 DYNAMIC KEYING SCHEMES

Dynamic keying schemes go through the phase of rekeying either periodically or on demand as needed by the network to refresh the security of the system. With rekeying, the sensors dynamically exchange keys that are used for securing the communication.

A dynamic en route filtering mechanism that does not exchange explicit control messages for rekeying. Provision of one-time keys for each packet transmitted to avoid stale keys . A Modular and flexible security architecture with a simple technique for

ensuring authenticity, integrity, and no repudiation of data without enlarging packets with MACs and a robust secure communication framework that is operational in direct communication situations and over unreliable medium access control layers.

2.3 RC4 MECHANISM

RC4 or Rivest Code 4 is the most widely-used software stream cipher and is used in popular protocols such as Secure Sockets Layer (SSL) (to protect Internet traffic) and WEP (to secure wireless networks). Our system employs a simple encoding process, which is essentially the process of permutation of the bits in the packet according to the dynamically created permutation code generated via RC4. Specifically, each sensed data is protected using a simple encoding scheme based on a permutation code generated with the RC4 encryption scheme and sent toward the sink.

2.4 STATES OF SENSOR NODES

After deployment, sensor nodes traverse several functional states. The states mainly include node-stay-alive, packet reception, transmission, encoding, and decoding. As each of these actions occur, the virtual energy in a sensor node is depleted.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

3.1 HARDWARE SPECIFICATIONS:

- Processor : Pentium IV
- Hard Disk : 40 GB
- Floppy Drive : 1.44 Mb
- Ram : 256 Mb

3.2 SOFTWARE SPECIFICATIONS:

- Operating system : - Windows XP Professional.
- Front End : - Visual Studio.Net 2005
- Coding Language : - Visual C# .Net

3.2 .1 LANGUAGE SPECIFICATIONS:

3.2 .2 INTRODUCTION TO .NET

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There's no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate.

2.3 THE .NET FRAMEWORK



The .NET Framework has two main parts:

1. The Common Language Runtime (CLR).
2. A hierarchical set of class libraries.

The CLR is described as the “execution engine” of .NET. It provides the environment within which programs run. The most important features are

- Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
- Memory management, notably including garbage collection.
- Checking and enforcing security restrictions on the running code.

MANAGED CODE

The code that targets .NET, and which contains certain extra Information - metadata” - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.

MANAGED DATA

With Managed Code comes Managed Data. CLR provides memory allocation and Deal location facilities, and garbage collection. Some .NET languages use

Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you're using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn't get garbage collected but instead is looked after by unmanaged code.

COMMON TYPE SYSTEM

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling.

COMMON LANGUAGE SPECIFICATION

The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

THE CLASS LIBRARY

DOT NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte,

Double, Boolean, and String, as well as Object. All objects derive from System.Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of converting value types to object types if and when necessary.

3.2.5 LANGUAGES SUPPORTED BY .NET

The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft's old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading.

Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in C#. Microsoft's new language. It's a C-style language that is essentially "C++ for Rapid Application Development". Unlike other languages, its specification is just the grammar of the language. It has no standard library of its own, and instead has been designed with the intention of using the .NET libraries as its own.

Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State's Perl Dev Kit.

C#.NET is also compliant with CLS (Common Language Specification) and also supports structured exception handling. CLS is set of rules and constructs that are well supported by the CLR (Common Language Runtime). CLR is the runtime environment provided by the .NET Framework. It manages the execution of the code and also makes the development process easier by providing services.

C#.NET is a CLS-compliant language. Any objects, classes, or components that created in C#.NET can be used in any other CLS-compliant language. In addition, we can use objects, classes, and components created in other CLS-compliant languages in C#.NET .The use of CLS ensures complete interoperability among applications, regardless of the languages used to create the application.

CONSTRUCTORS AND DESTRUCTORS:

Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words, destructors are used to release the resources allocated to the object. In C#.NET the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize procedure can be called only from the class it belongs to or from derived classes.

GARBAGE COLLECTION

Garbage Collection is another new feature in C#.NET. The .NET Framework monitors allocated resources, such as objects and variables. In addition, the .NET Framework automatically releases memory for reuse by destroying objects that are no longer in use.

In C#.NET, the garbage collector checks for the objects that are not currently in use by applications. When the garbage collector comes across an object that is marked for garbage collection, it releases the memory occupied by the object.

OVERLOADING

Overloading is another feature in C#. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of rules or arguments. Besides using overloading for procedures, we can use it for constructors and properties in a class.

3.3 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During any system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to the high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 4

SYSTEM DESIGN

4.1 OVERALL ARCHITECTURE

This overall architecture depicts the general workflow of the proposed system. It contains the modules through which the data flow take place. The secret key and the file is given as input in server side. The key is generated accordingly, after which IP authentication takes place and request stage is complete thereafter. Then the file transfer takes place.

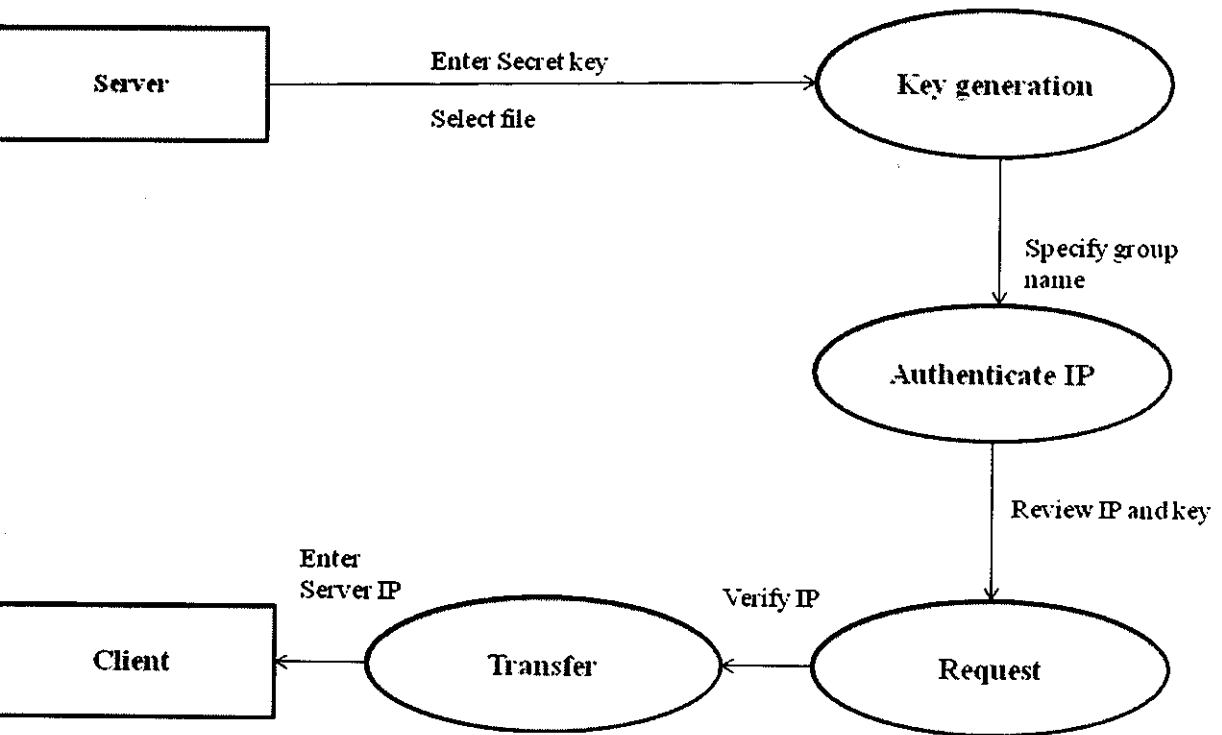


Fig 4.1:Architecture diagram

4.2 UML REPRESENTATION

4.2.1 USE CASE DIAGRAM

A Use case diagram in the Unified Modeling Language is a type of behavioral diagram defined by and created for use-case analysis. Here the user's role is depicted. The process performed by the user are displayed in use case

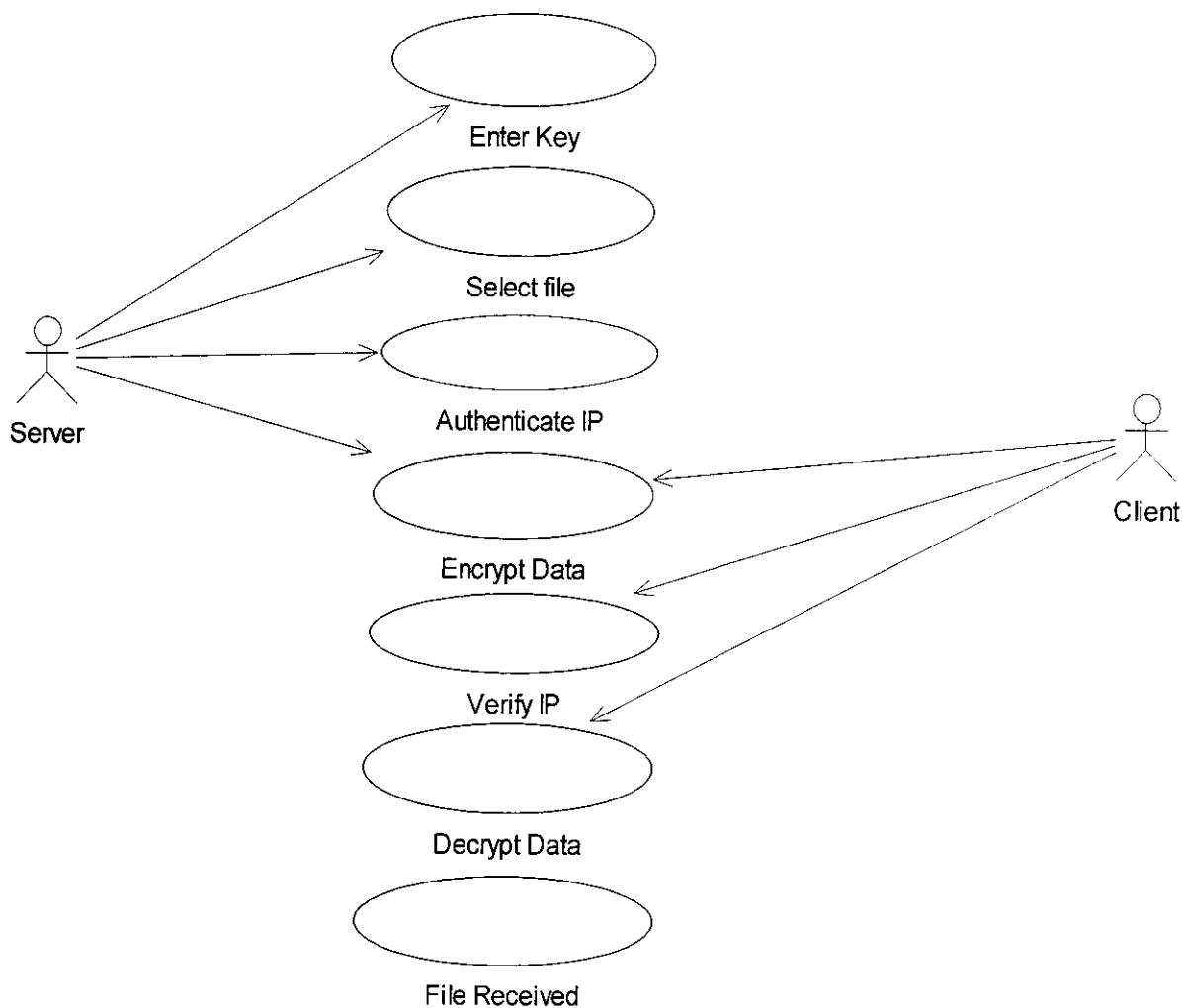


Fig 4.2: Use Case Diagram

4.2.2 SEQUENCE DIAGRAM

A Sequence diagram in unified modeling language is a kind of interaction diagram that shows how processes operate with one another. Sequence diagrams are also called as event diagrams, event scenarios and timing diagram. Here the sequence gets initiated with login process and in final phase extraction is done.

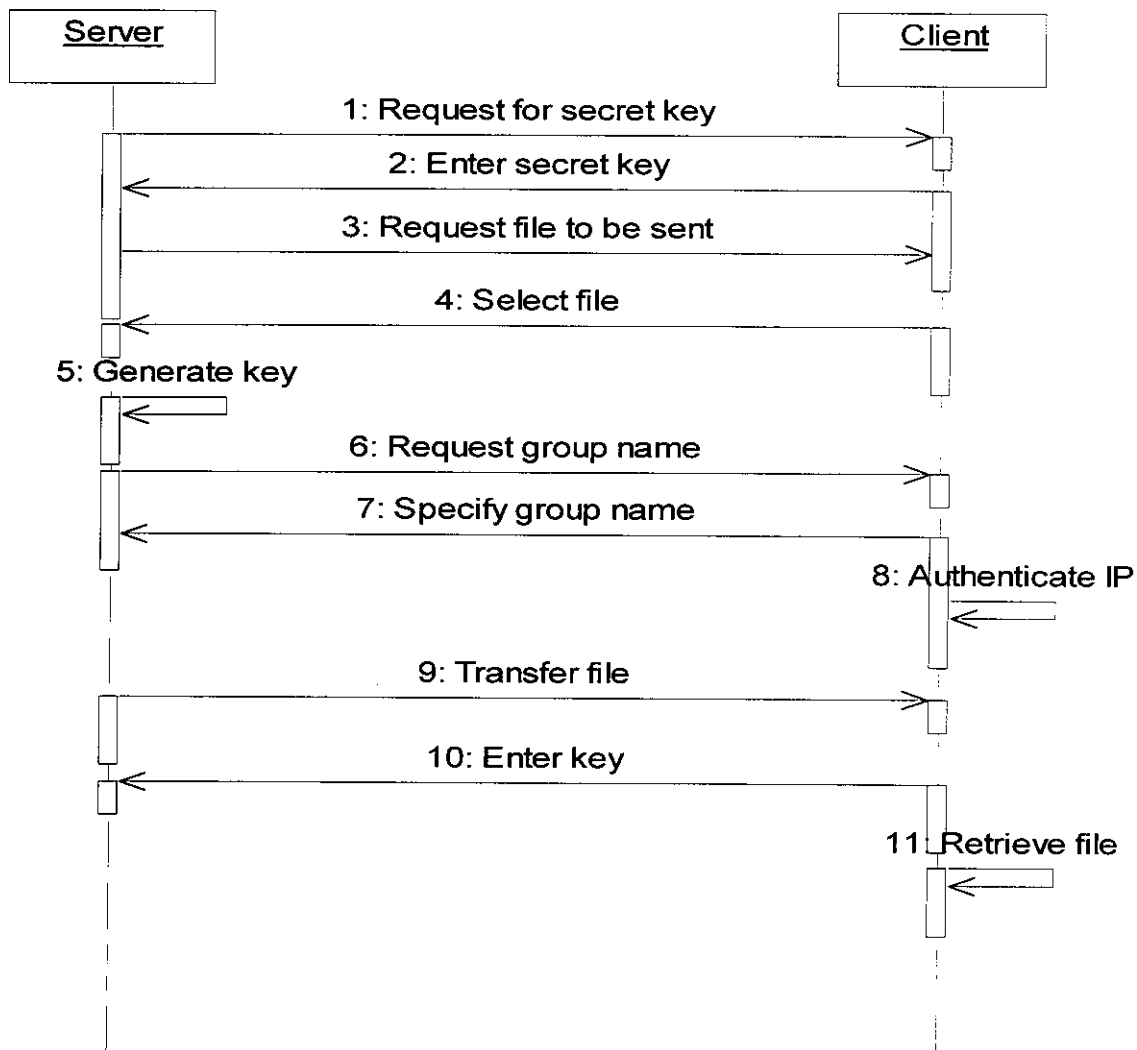


Fig 4.3: Sequence Diagram

CHAPTER 5

SYSTEM DESCRIPTION

5.1 NETWORKING MODULE

- Provides the method to generate the key using the secret key value input by the user.
- We establish a connection between the server and the client, by selecting the appropriate one from the workgroup specified.

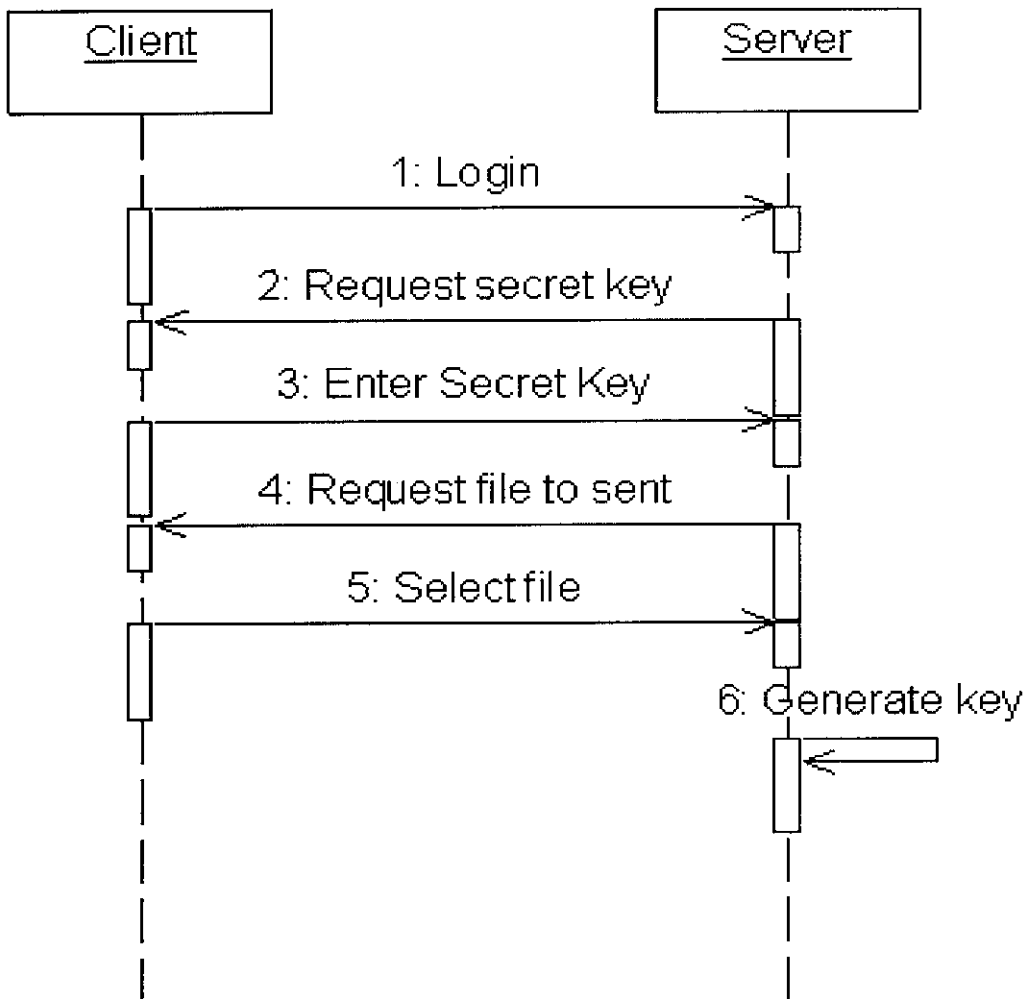


Fig 5.1:Sequence Diagram For Networking Module

5.2 STREAM CHIRPING MODULE

- At the transmitting end, the key stream is XORed with the plaintext stream, yielding a cipher text stream.
- The receiver, having the same seed key, generates synchronously the same key stream..

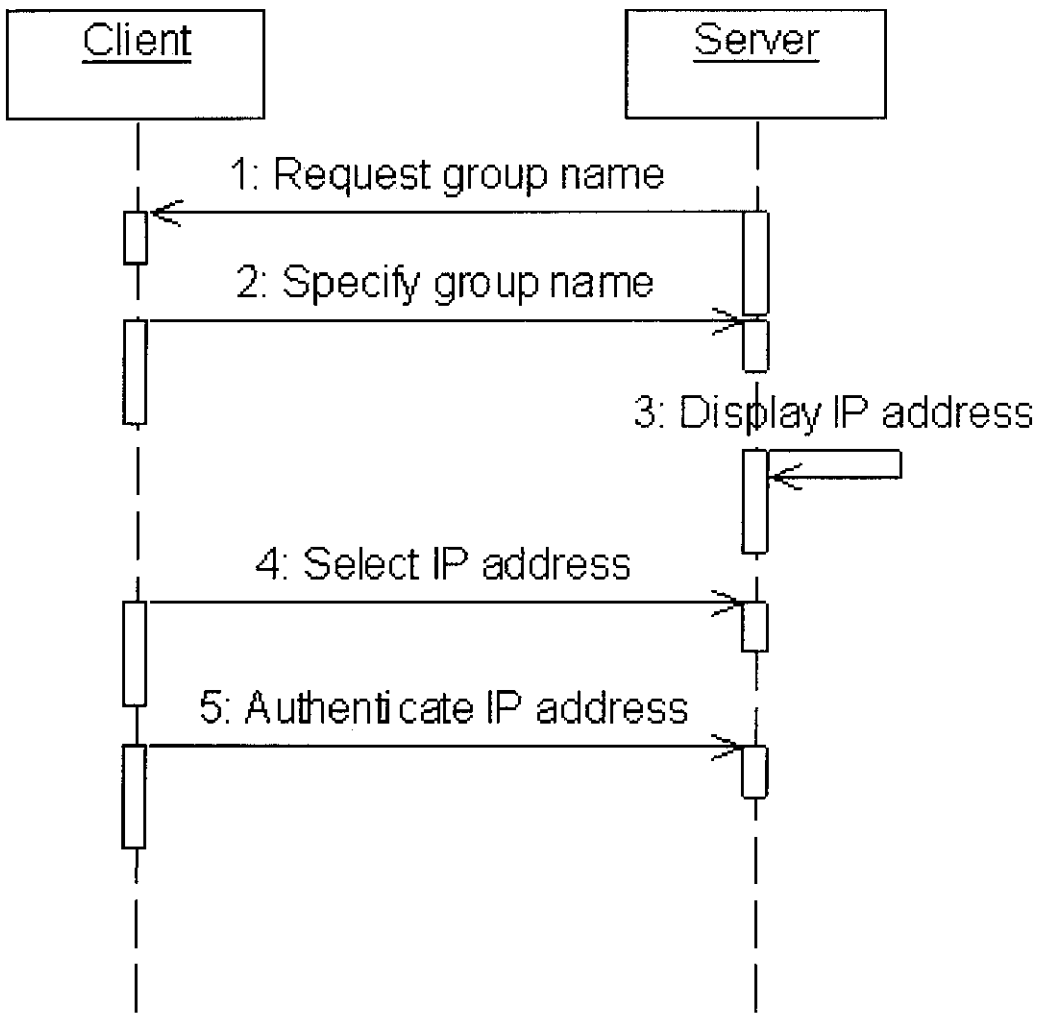


Fig 5.2:Sequence Diagram For Stream Chirping Module

5.3 RC4 MECHANISM

- RC4 is a software stream cipher and is used in popular protocols such as SSL and WEP.
- It provides a faster processing speed compared to other schemes.

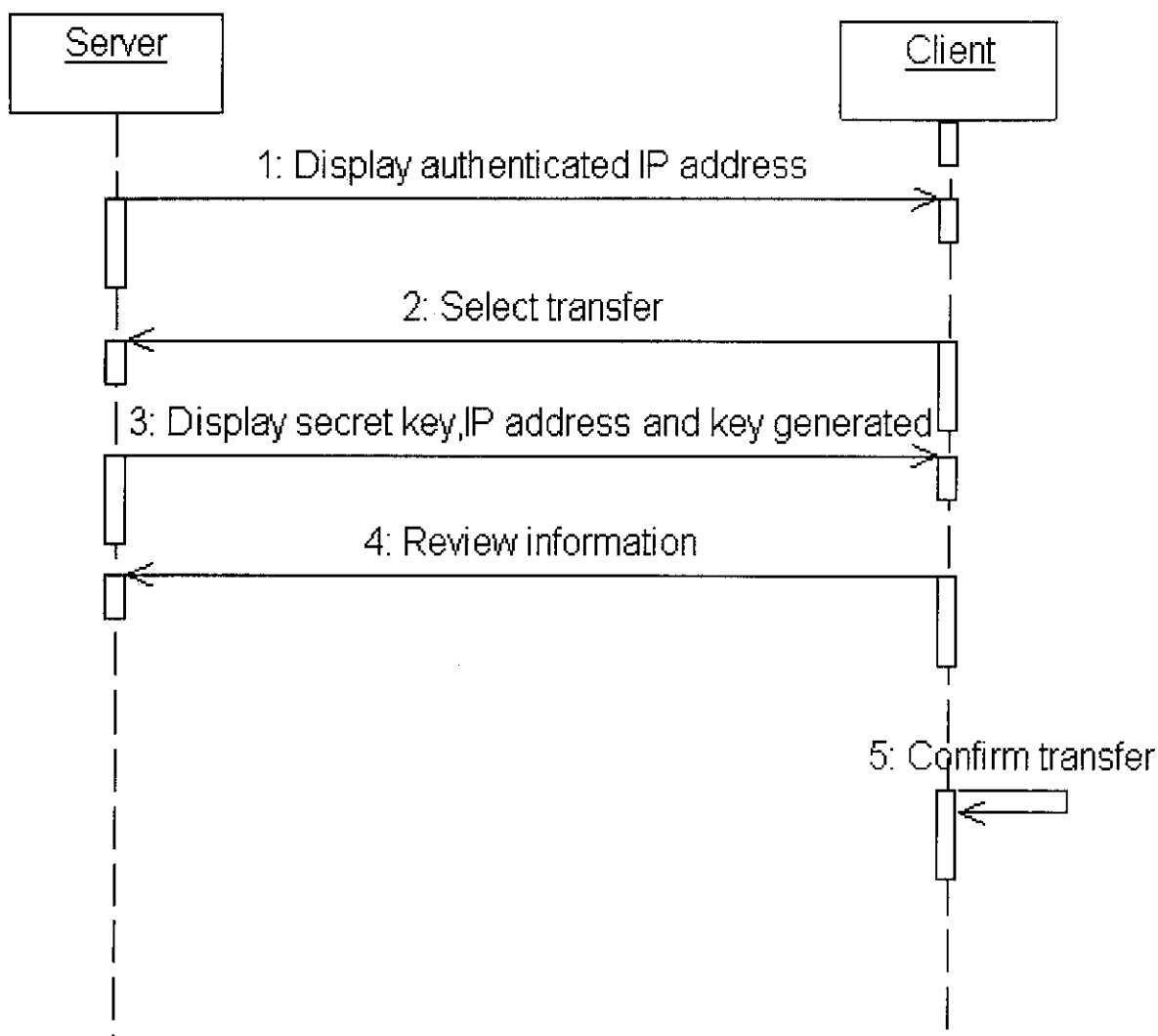


Fig 5.3:Sequence Diagram For Rc4 Mechanism

5.4 AUTHENTICATION MODULE

- Message authentication code is used in this system for the authentication purpose. $MAC(M, K)$ is a one-way transformation of the message m and a secret key k .
- Hash message authentication code (HMAC) is a hash transformation parameterized with a secret key.

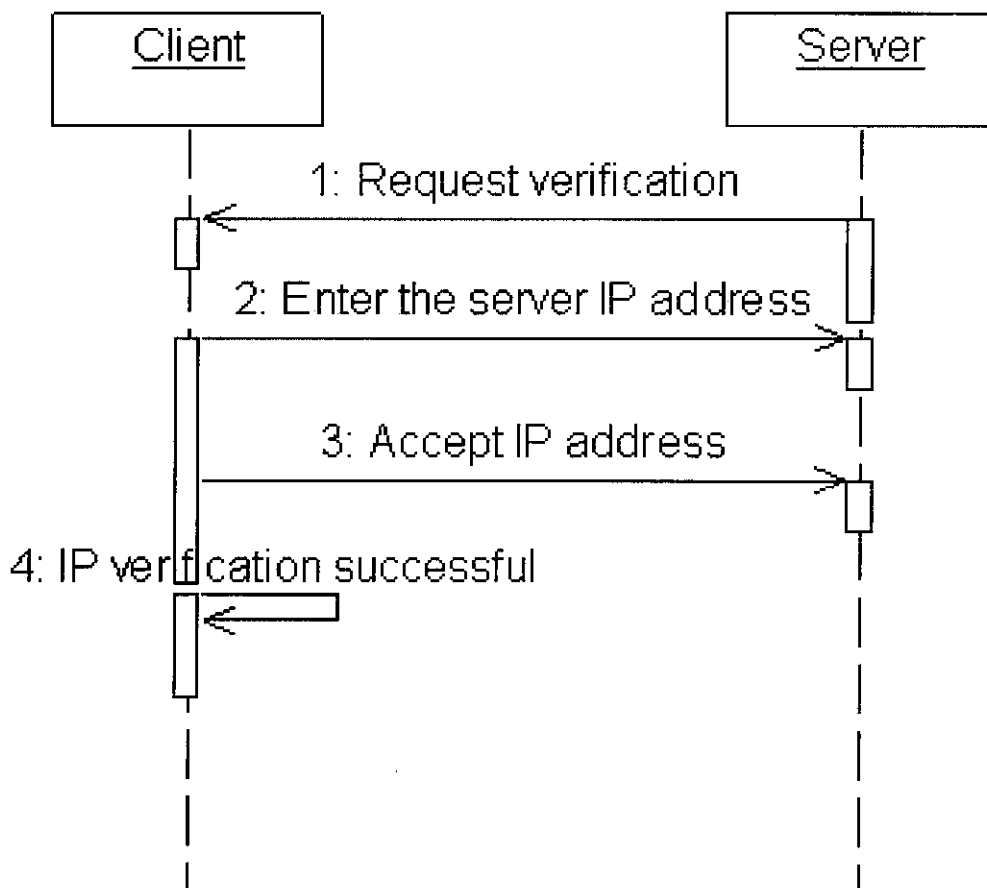


Fig 5.4: Sequence Diagram For Authentication Module

5.5 KEY RECOVERY MODULE

- At the client side, the key is recovered using the inverse method of the RC4 key generation i.e. the user needs to enter the same key as used in the server side to receive the file.
- Though the file has been received, the data cannot be extracted without entering the exact key.

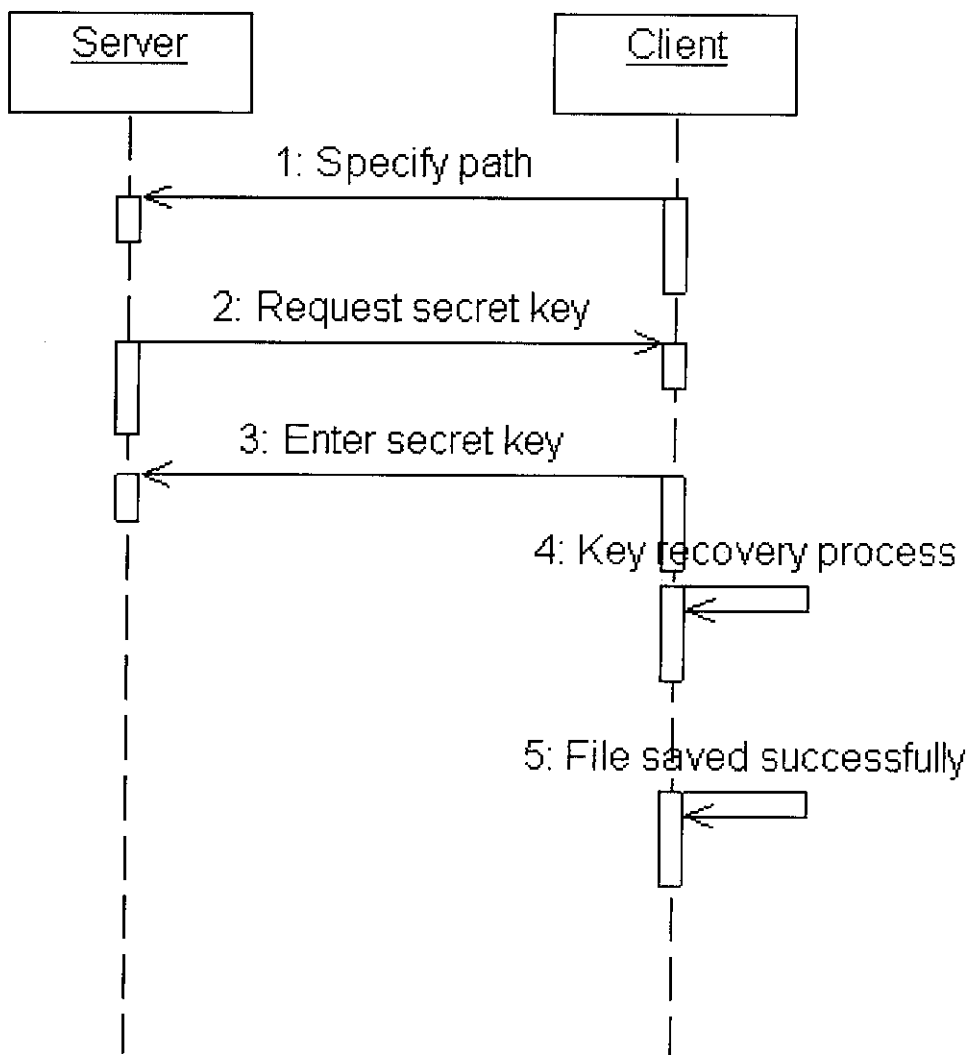


Fig 5.5:Sequence Diagram For Key Recovery Module

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 NETWORKING MODULE

This snapshot shows the system main interface of the system. Provides the method to generate the key using the secret key value input by the user. We establish a connection between the server and the client, by selecting the appropriate one from the workgroup specified.

CODING:

```
class ServerCode
{
    IPEndPoint ipEnd;
    Socket sock;
    string locsysip;
    public ServerCode()
    {
        IPEndPoint ipEntry = Dns.GetHostEntry(Environment.MachineName);
        IPAddress IpAddr = ipEntry.AddressList[0];
        locsysip = IpAddr.ToString();
        ipEnd = new IPEndPoint(IpAddr, 55);
        sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.IP);
        sock.Bind(ipEnd);
    }
}
```

```

public static string receivedPath;

public static string curMsg = "";

public static string reply = "";

public static int res;

public void StartServer()
{
    try
    {

        sock.Listen(100);

        Socket clientSock = sock.Accept();

        byte[] clientData = new byte[1024 * 5000];

        int receivedBytesLen = clientSock.Receive(clientData);

        string replymsg = Encoding.ASCII.GetString(clientData, 0, receivedBytesLen);

        if (replymsg == "Yes")
        {
            MessageBox.Show("Client Accepted.", "Message", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
        }

        else if (replymsg == "No")

```

```
{
    MessageBox.Show("Client Denied.", "Message", MessageBoxButtons.OK,
MessageBoxIcon.Information);
}

clientSock.Close();

StartServer();

}
catch (Exception ex)
{
    curMsg = "File Receiving error.";
}
}
}
```

SNAPSHOT

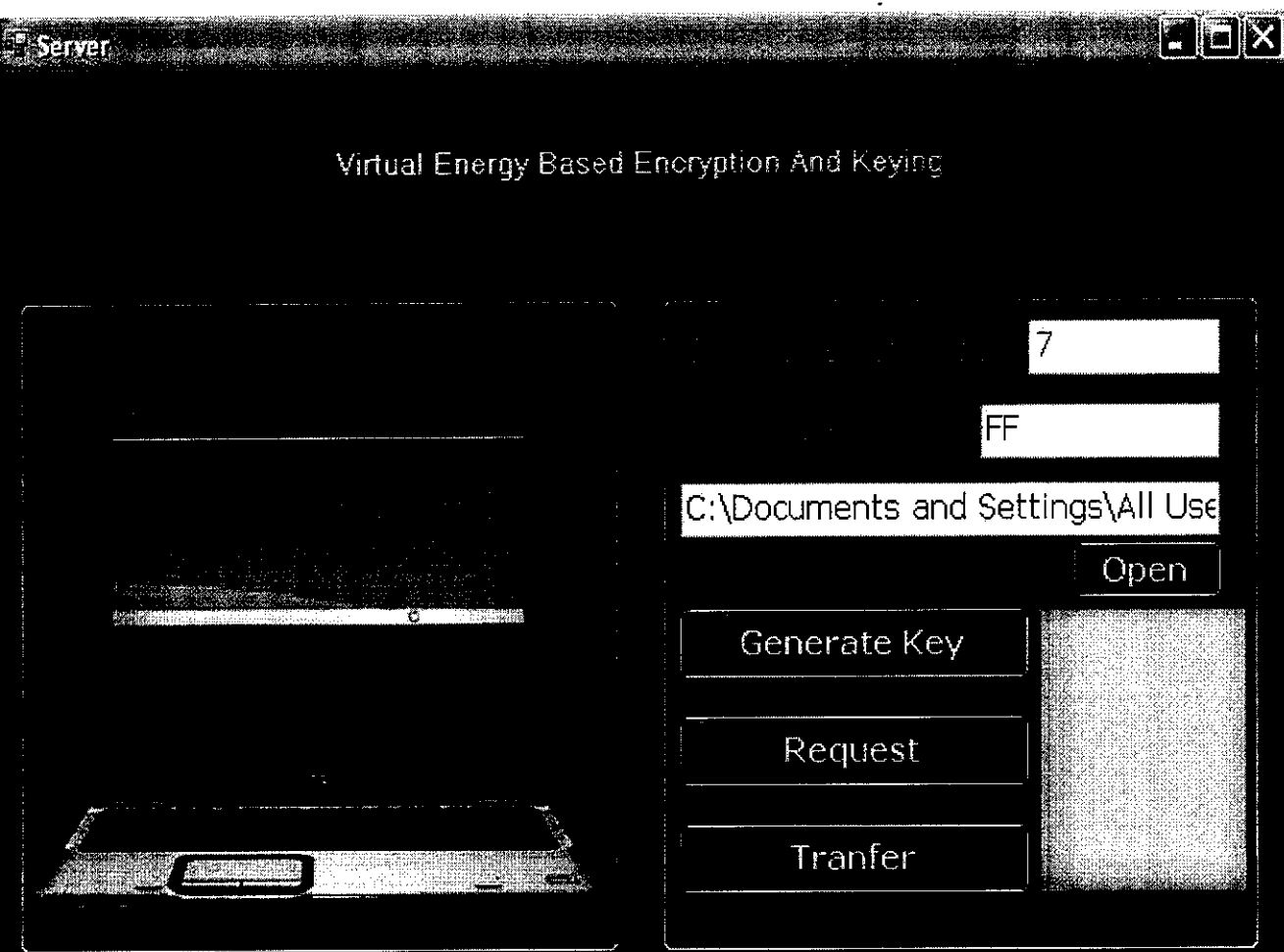


Fig 6.1: Snapshot Of System Main Page

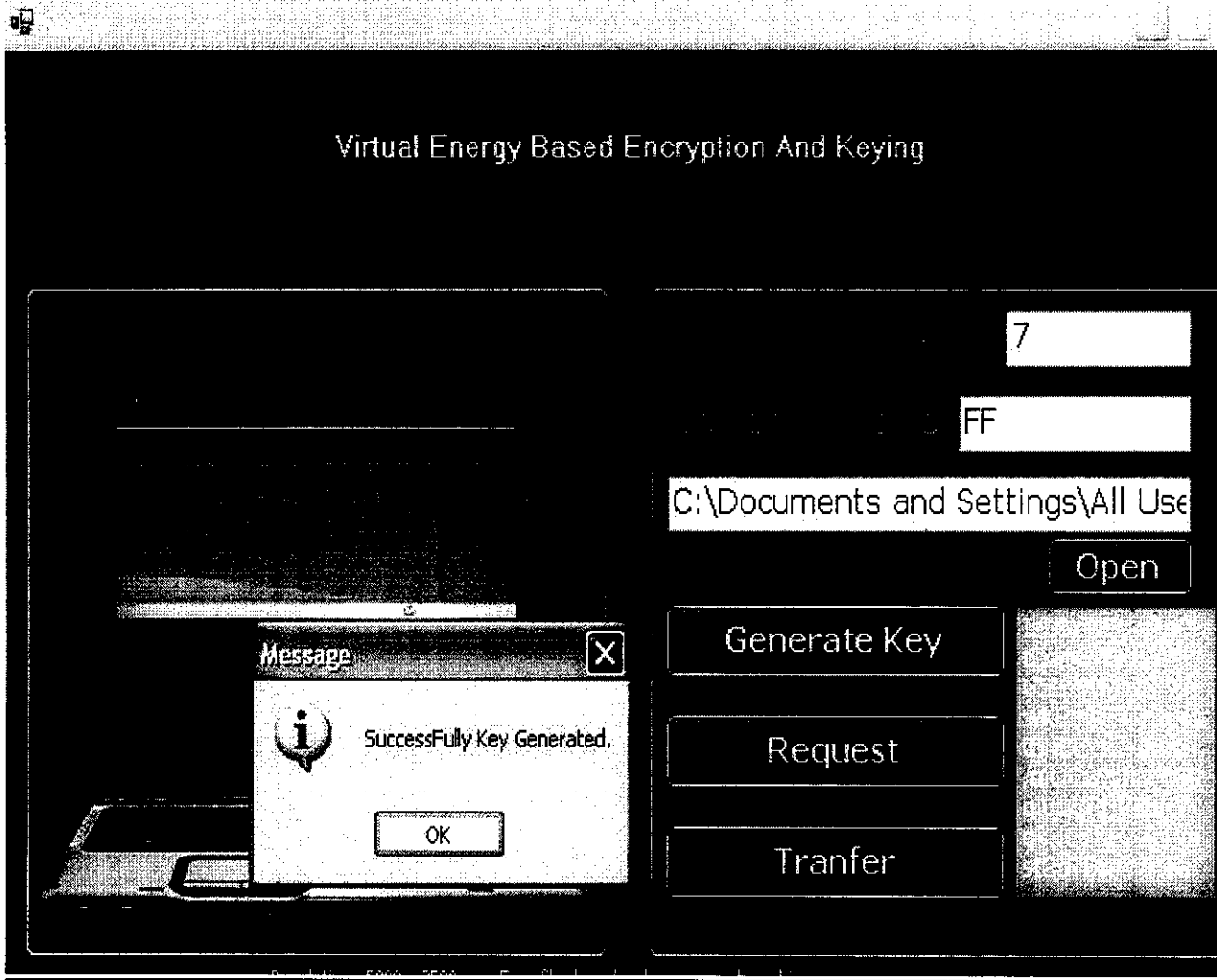


Fig 6.2:Snapshot Of System After Key Generation

6.2 STREAM CHIRPING MODULE

This snapshot shows the user input for group name to obtain IP address .At the transmitting end, the key stream is XORed with the plaintext stream, yielding a cipher

text stream. The receiver, having the same seed key, generates synchronously the same key stream.

CODING

```
public class CryptorEngine
{
    /// <summary>
    /// Encrypt a string using dual encryption method. Return a encrypted cipher Text
    /// </summary>
    /// <param name="toEncrypt">string to be encrypted</param>
    /// <param name="useHashing">use hashing? send to for extra security</param>
    /// <returns></returns>
    ///
    public static string SValuKey="";

    public static string Key
    {
        get
        {
            return SValuKey;
        }
        set
        {
            SValuKey = value;
        }
    }
}
```



```

public static string Encrypt(string toEncrypt, bool useHashing)
{
    byte[] keyArray;

    byte[] toEncryptArray = UTF8Encoding.UTF8.GetBytes(toEncrypt);

    System.Configuration.AppSettingsReader settingsReader = new AppSettingsReader();

    // Get the key from config file
    string key = Key.Trim();// (string)settingsReader.GetValue("Syed Moshiur Murshed", typeof(String));

    //System.Windows.Forms.MessageBox.Show(key);

    if (useHashing)
    {
        MD5CryptoServiceProvider hashmd5 = new MD5CryptoServiceProvider();
        keyArray = hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));
        hashmd5.Clear();
    }
    else
        keyArray = UTF8Encoding.UTF8.GetBytes(key);

    TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();
    tdes.Key = keyArray;
    tdes.Mode = CipherMode.ECB;
    tdes.Padding = PaddingMode.PKCS7;

    ICryptoTransform cTransform = tdes.CreateEncryptor();
    byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);
    tdes.Clear();
}

```

```

return Convert.ToBase64String(resultArray, 0, resultArray.Length);
}
/// <summary>
/// DeCrypt a string using dual encryption method. Return a DeCrypted clear string
/// </summary>
/// <param name="cipherString">encrypted string</param>
/// <param name="useHashing">Did you use hashing to encrypt this data? pass true is yes</param>
/// <returns></returns>
public static string Decrypt(string cipherString, bool useHashing)
{
    byte[] keyArray;
    byte[] toEncryptArray = Convert.FromBase64String(cipherString);

    System.Configuration.AppSettingsReader settingsReader = new AppSettingsReader();
    //Get your key from config file to open the lock!
    string key = Key.Trim();// (string)settingsReader.GetValue("Syed Moshiur Murshed", typeof(String));

    if (useHashing)
    {
        MD5CryptoServiceProvider hashmd5 = new MD5CryptoServiceProvider();
        keyArray = hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));
        hashmd5.Clear();
    }
    else
        keyArray = UTF8Encoding.UTF8.GetBytes(key);
}

```

```
TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();
tdes.Key = keyArray;
tdes.Mode = CipherMode.ECB;
tdes.Padding = PaddingMode.PKCS7;

ICryptoTransform cTransform = tdes.CreateDecryptor();
byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);

tdes.Clear();
return UTF8Encoding.UTF8.GetString(resultArray);
}
}
```

SNAPSHOT:

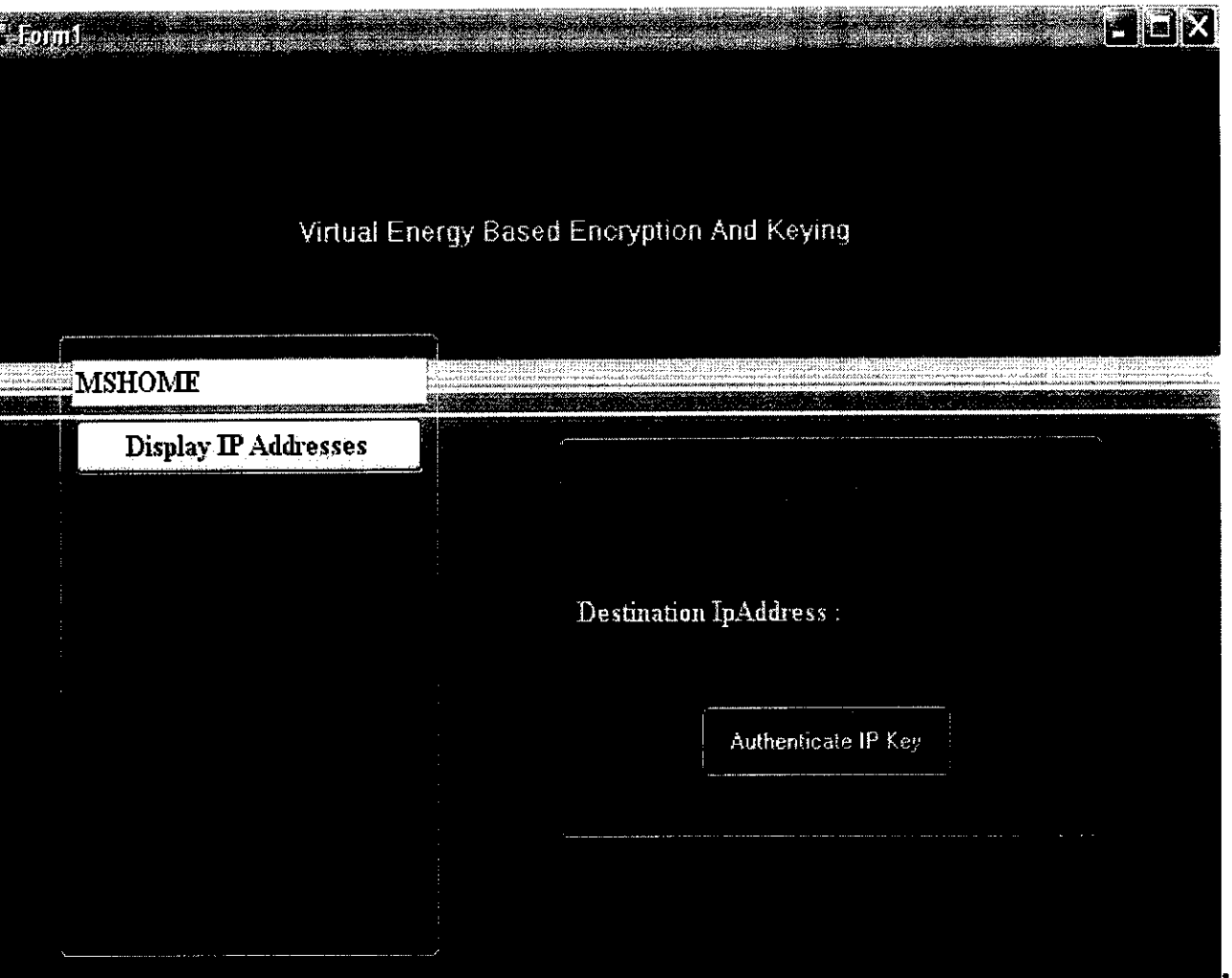


Fig 6.3:Snapshot For Group Name Input

Virtual Energy Based Encryption And Keying

MSHOME

Display IP Addresses

```
COMPUTER3      192.168.1.30
ENDEAVOU-2FE4B192.168.1.2
ENDEAVOU-6D4D9192.168.1.39
ENDEAVOU-D2428192.168.1.22
ENDEAVOU-PC    192.168.1.28
```

Destination IpAddress : 192.168.1.22

Authenticate IP Key

Fig 6.4 :Snapshot For IP List And IP Authentication

3 RC4 MECHANISM

This snapshot shows the selected IP address and its ready to be transferred to the client system. RC4 is a software stream cipher and is used in popular protocols such as SSL and WEP. It provides a faster processing speed compared to other schemes.

CODING

```
public partial class IPVerify : Form
{
    public IPVerify()
    {
        InitializeComponent();
    }

    public string ipaddress;

    private void btnCancel_Click(object sender, EventArgs e)
    {

    }

    private void btnAccept_Click(object sender, EventArgs e)
    {

    }

    public void send(byte[] data)
    {
```

```

try
{
    IPAddress[] ipAddress = Dns.GetHostAddresses(txtServerIp.Text);
    IPEndPoint ipEnd = new IPEndPoint(ipAddress[0], 55);
    Socket clientSock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
protocolType.IP);

    byte[] replyByte = Encoding.ASCII.GetBytes("Yes");

    clientSock.Connect(ipEnd);
    System.Threading.Thread.Sleep(1000);
    clientSock.Send(replyByte);
    clientSock.Close();
}

catch (Exception ex)
{
    if (ex.Message == "A connection attempt failed because the connected party did not properly
respond after a period of time, or established connection failed because connected host has failed to respond")
    {

        MessageBox.Show("No Such System Available Try other IP");
    }
else
{
    if (ex.Message == "No connection could be made because the target machine actively refused it")

```

```
{
    MessageBox.Show("File Sending fail. Because server not running.");
}
else
{
    MessageBox.Show("File Sending fail." + ex.Message);
}
}
}
}
```

```
private void IPVerify_Load(object sender, EventArgs e)
```

```
{
    txtServerIp.Text = ipaddress;
}
```

```
private void btnCancel_Click_1(object sender, EventArgs e)
```

```
{
    byte[] tmpdy = Encoding.ASCII.GetBytes("No");
    send(tmpdy);
    Close();
}
```

```
private void btnAccept_Click_1(object sender, EventArgs e)
```

```
{
    Client.accept = "Accepted";
}
```



```
byte[] tmpac = Encoding.ASCII.GetBytes("Yes");
```

```
send(tmpac);
```

```
Close();
```

```
}
```

```
}
```

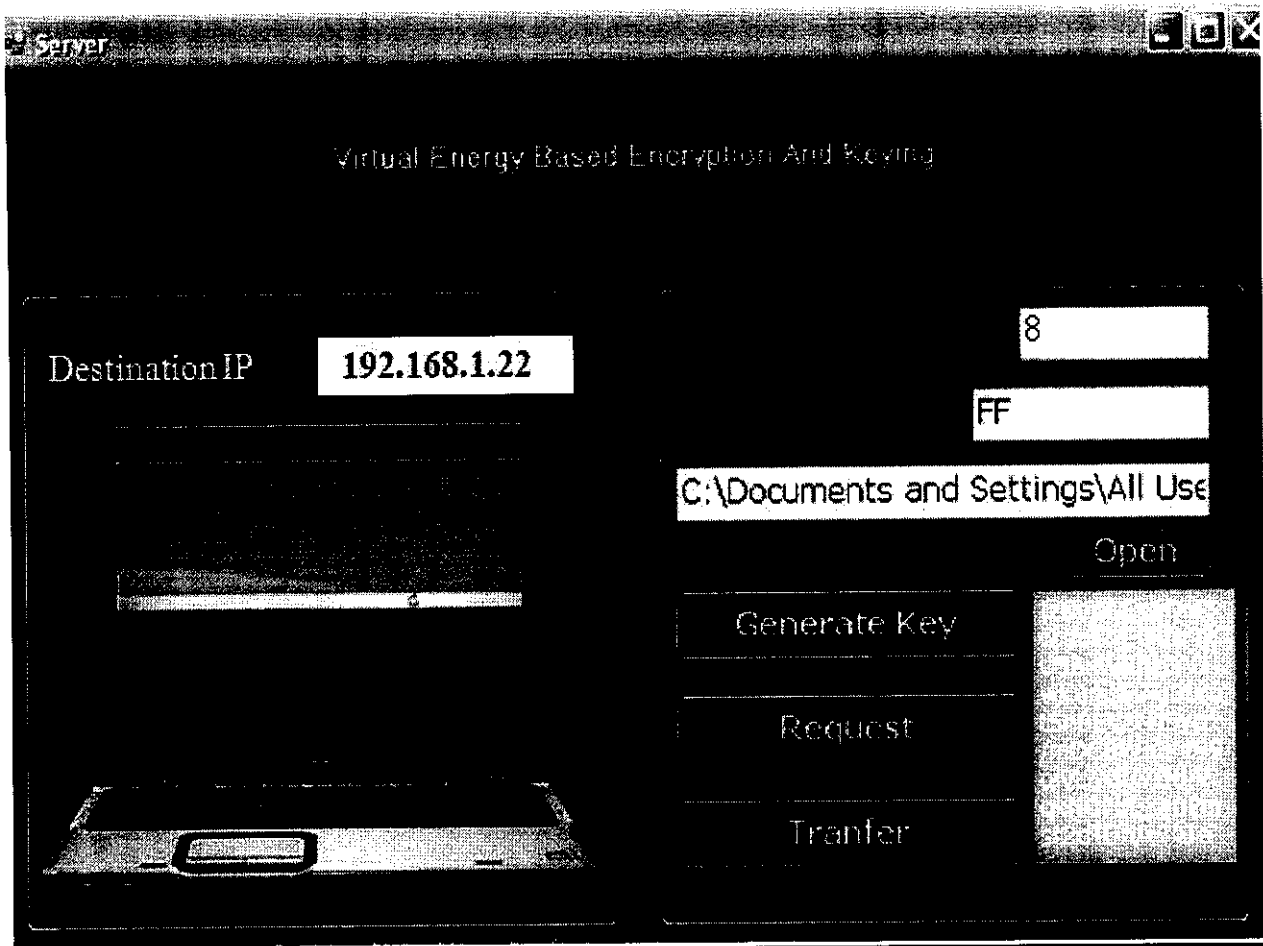


Fig 6.5: Snapshot For Transfer Operation

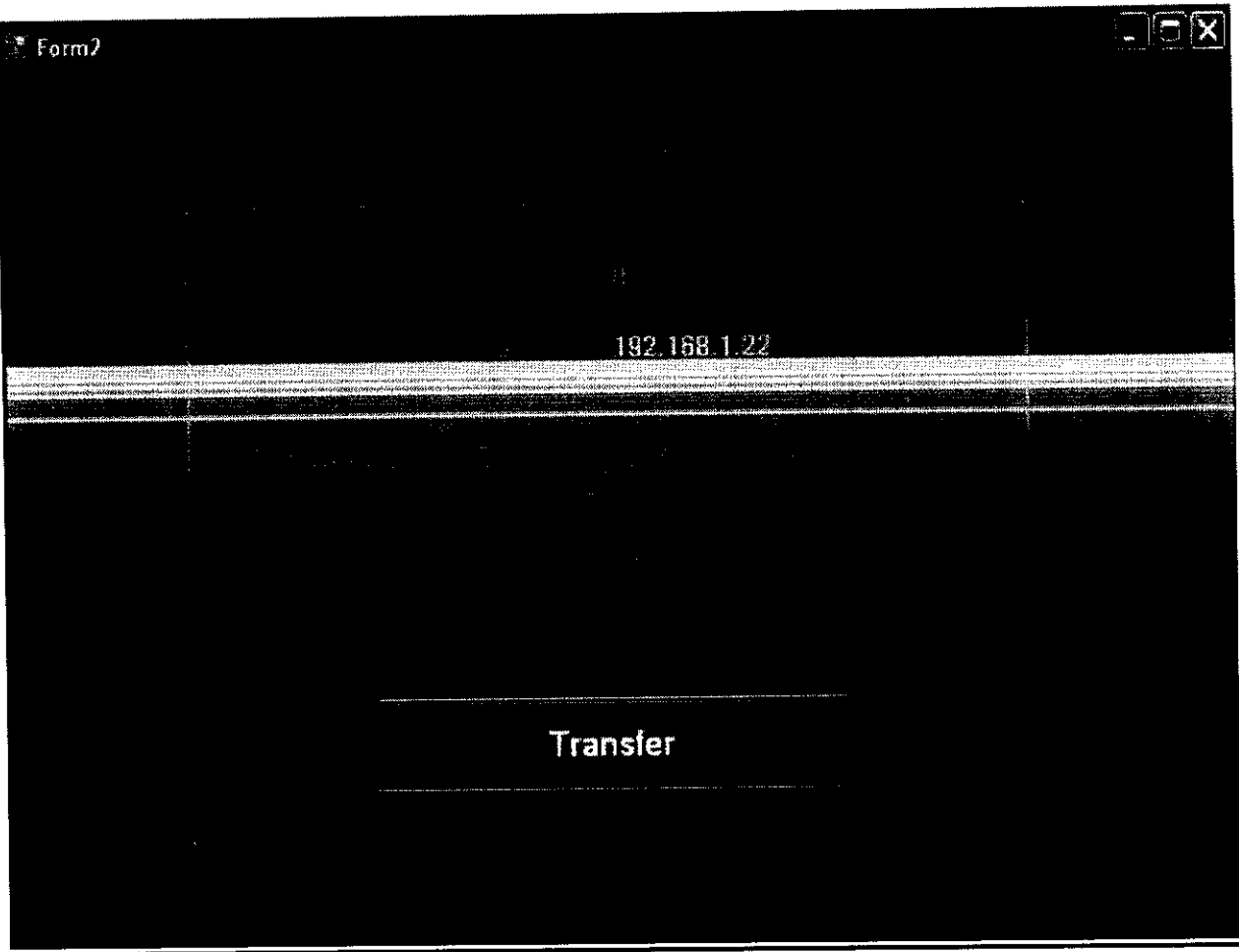


Fig 6.6: Snapshot For Summary And Transfer Operation

5.4 AUTHENTICATION MODULE

This snapshot shows the client system awaiting for user to verify the IP address of the sender. Message authentication code is used in this system for the authentication purpose. MAC(M K) is a one-way transformation of the message m and a secret key k. Hash message authentication code (HMAC) is a hash transformation parameterized with a secret key.

CODING

```
public partial class Authenticate : Form
{
    public Authenticate()
    {
        InitializeComponent();
    }

    byte[] senddata;

    private void label4_Click(object sender, EventArgs e)
    {

    }

    private void Authenticate_Load(object sender, EventArgs e)
    {

};
```

```

}

public void send(byte[] data)
{
    try
    {
        IPAddress[] ipAddress = Dns.GetHostAddresses(lblDestIP.Text );
        IPEndPoint ipEnd = new IPEndPoint(ipAddress[0], 56);
        Socket clientSock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.IP);

        // byte[] replyByte = Encoding.ASCII.GetBytes("Request");

        clientSock.Connect(ipEnd);
        System.Threading.Thread.Sleep(1000);
        clientSock.Send(data);
        clientSock.Close();
    }

    catch (Exception ex)
    {
        if (ex.Message == "A connection attempt failed because the connected party did not properly
respond after a period of time, or established connection failed because connected host has failed to respond")
        {

```

```
    MessageBox.Show("No Such System Available Try other IP");
}
else
{
    if (ex.Message == "No connection could be made because the target machine actively refused it")
    {
        MessageBox.Show("File Sending fail. Because server not running.");
    }
    else
    {
        MessageBox.Show("File Sending fail." + ex.Message);
    }
}
}
}
```

```
private void btnAuthenticate_Click(object sender, EventArgs e)
{
    senddata = Encoding.ASCII.GetBytes(ipFetcher1.IpAddress);
    send(senddata);
    this.Close();
}
```

```
private void ipFetcher1_Click(object sender, EventArgs e)
{
```

```
}
```

```
private void ipFetcher1_MouseMove(object sender, MouseEventArgs e)
```

```
{
```

```
    lblDestIP.Text = ipFetcher1.IpAddress;
```

```
    front.DestinationIp = ipFetcher1.IpAddress;
```

SNAPSHOT

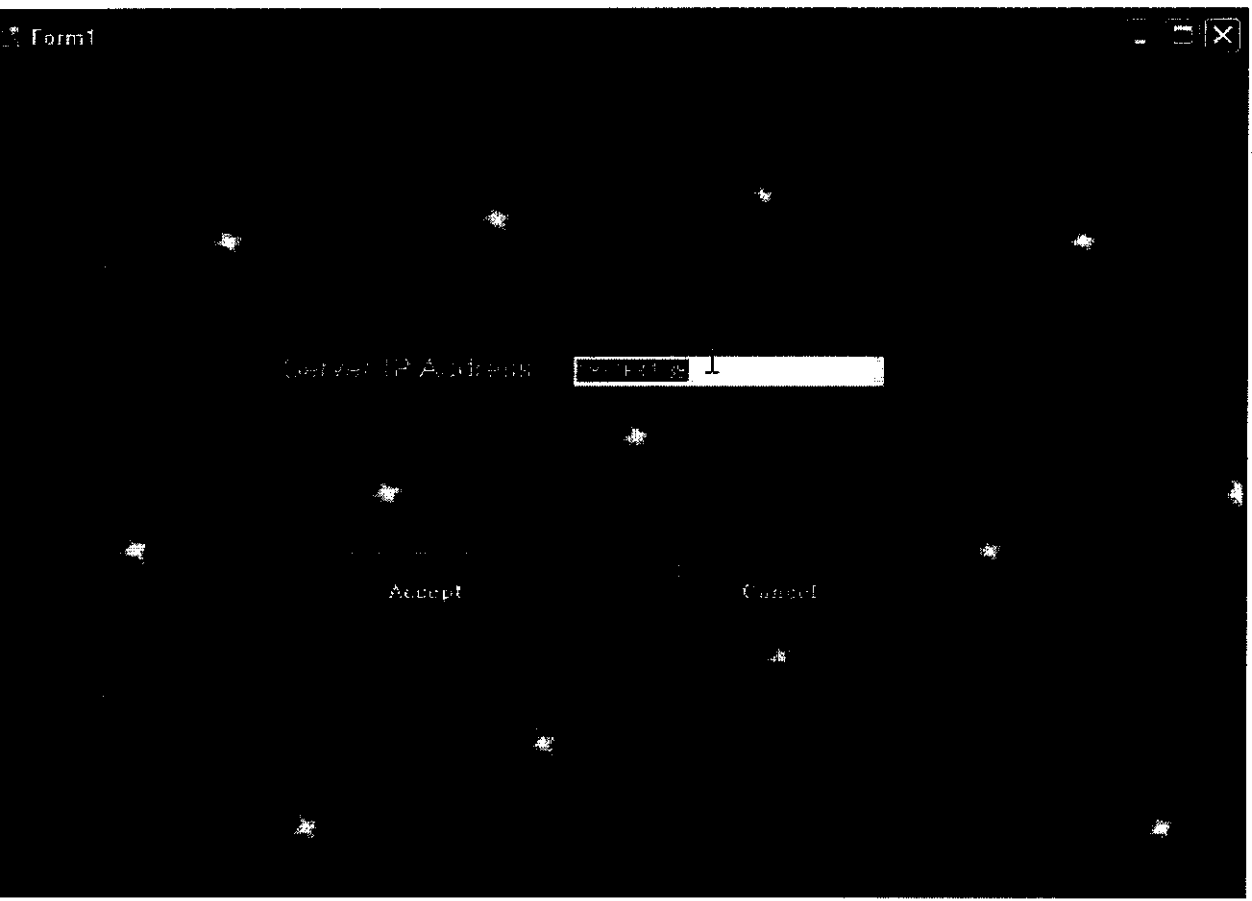


Fig 6.7: Snapshot For Client Ip Verification

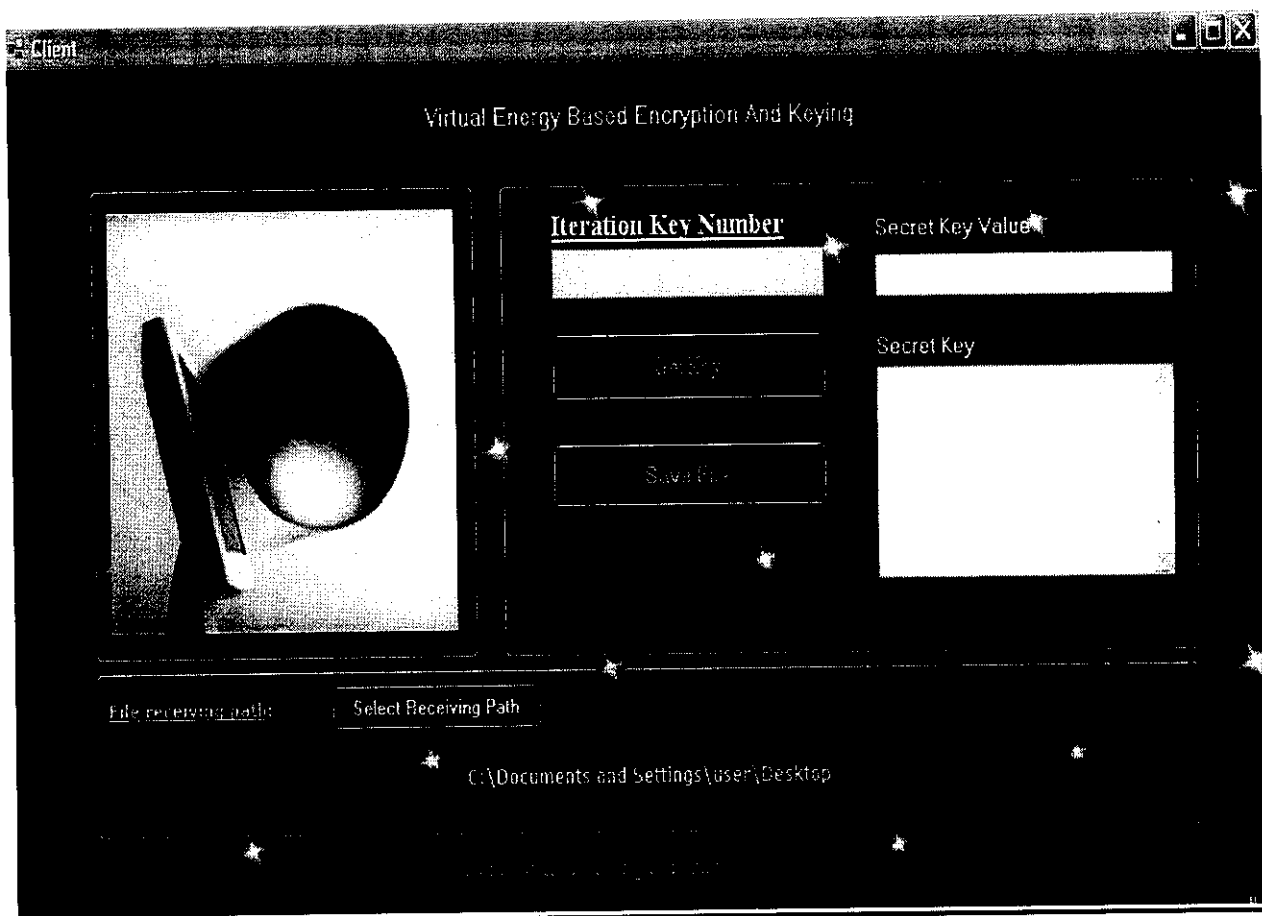


Fig 6.8: Snapshot For IP Verification Successful

5.5 KEY RECOVERY MODULE

This snapshot shows the client system where the user enters the iteration key number to recover the key. At the client side, the key is recovered using the inverse method of the RC4 key generation i.e. the user needs to enter the same key as used on the server side to receive the file. Though the file has been received, the data cannot be extracted without entering the exact key.

CODING

```
class Hmach
```

```
{  
  
    string ChiperTxt;  
  
    string DecTxt;  
  
    public String Encrypt(string EncValue, int IterationValue)  
    {  
        for (int i = 0; i <= IterationValue; i++)  
        {  
            string clearText = EncValue.Trim();  
            string cipherText = CryptorEngine.Encrypt(clearText, true);  
            ChiperTxt = cipherText;  
            EncValue = ChiperTxt;  
        }  
        return EncValue;  
    }  
  
    public string Decrypt(string DecValue, int IterationValue)  
    {  
        for (int i = 0; i <= IterationValue; i++)  
        {  
            string cipherText = DecValue.Trim();  
            string decryptedText = CryptorEngine.Decrypt(cipherText, true);  
            DecTxt = decryptedText;  
            DecValue = DecTxt;  
        }  
    }  
}
```

```
return DecValue;
```

```
}
```

SNAPSHOT

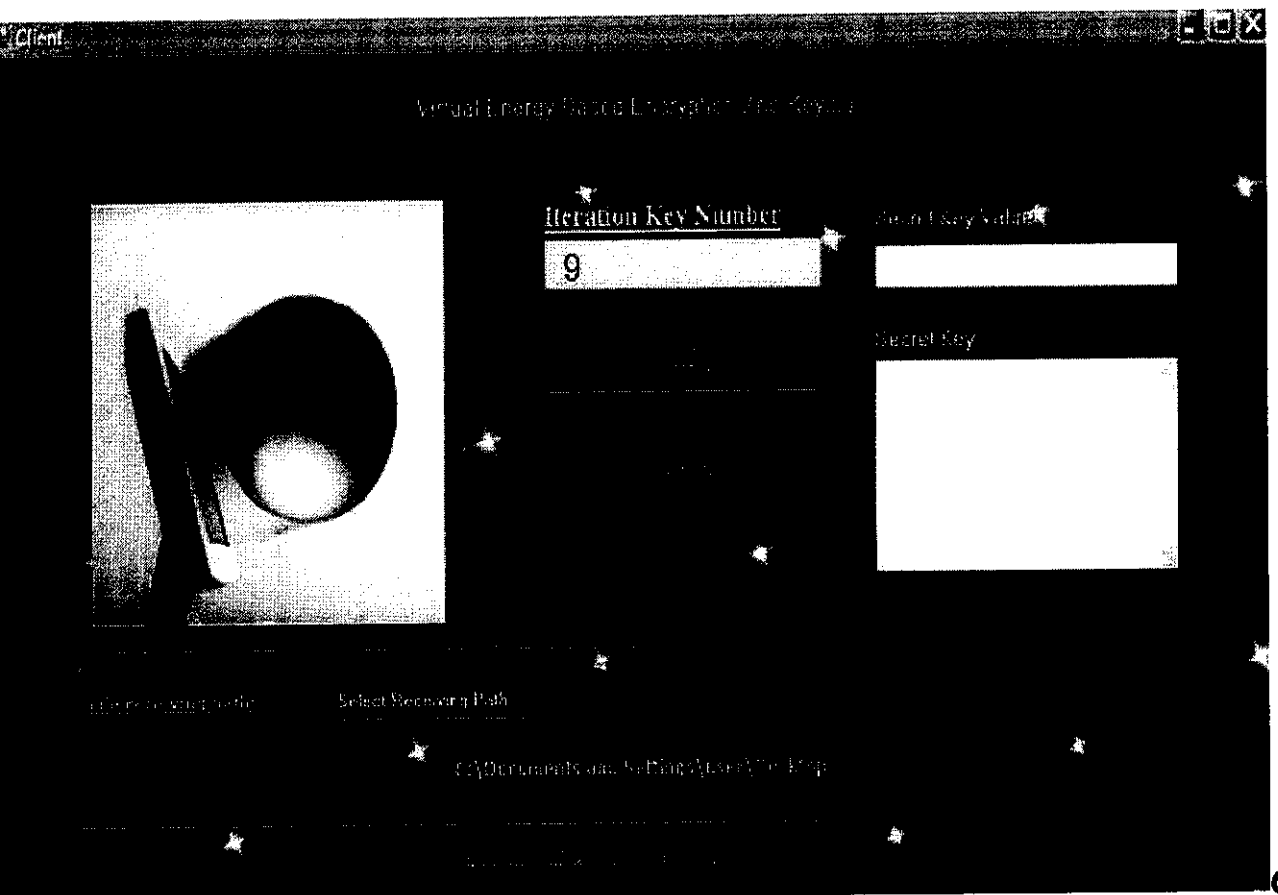


Fig 6.9: Snapshot For Iteration Key Number

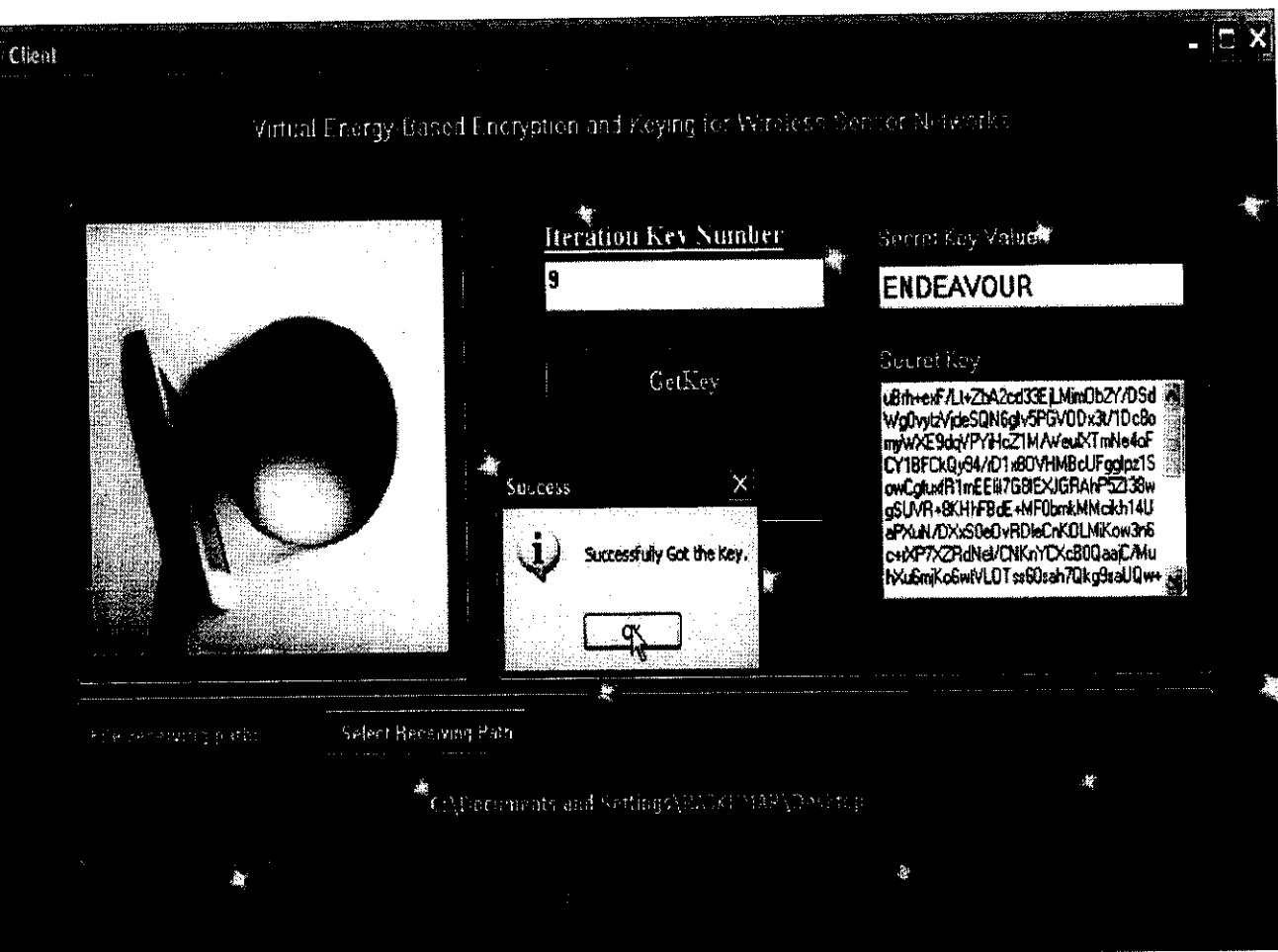


Fig 6.10: Snapshot For Key Recovery

CHAPTER 7

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.1 TESTING AND VARIOUS METHODOLOGIES

UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.



FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes

must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7.2 TEST CASES

“A test case has components that describe an input, action or event and an expected response, to determine if a feature of an application is working correctly.”

CASE GENERATION REPORT:

Test Type	Case	Expected Result	Observed Result
Operational, Unit, Functional Test	Key generation	To generate the key for the data to be sent,	The key is generated successfully for the data to be sent.
Functional Test	Authenticate IP address	To authenticate the selected IP address	The IP address is successfully authenticated.
Functional Test, Unit Test	Encryption	Receive the file to be encrypted and encrypt according to the key.	The input file is successfully encrypted.
Functional Test, Unit Test	Decryption	Receive the file to be decrypted and decrypt with same key.	The encrypted file is successfully decrypted.
Stress Test, Acceptance Test, Load Test	IP address Verification	To verify the IP address of the sender.	The sender's IP address is verified successfully.

Functional Test, Acceptance Test, Load Test	Retrieve	Receives the file and retrieve the encrypted file.	The file is retrieved successfully upon entering the key.
---	----------	--	---

TABLE 7.1: CASE GENERATION REPORT

All the above validations on buttons have been verified and they are successfully executed. The flow is tested at different possible conditions by means of this testing.

CHAPTER 8

FUTURE ENHANCEMENT

- Our system use a key called as manual key which serves as an effective and secure method to facilitate the way being the data is transferred.
- Here we use a secret key value in both server and client side to transfer data in a safe and efficient manner in wireless sensor networks.
- Our future work will address insider threats and dynamic paths which are a very common problem now days in sensor networks.
- Also we like to improve security by sending the key file separately on any other secured way.

CHAPTER 9

CONCLUSION

We have evaluated our system's feasibility and performance through both theoretical analysis and practical operation. Our results show that our system can provide optimal performance in a variety of network configurations depending largely on the application of the sensor network.

CHAPTER 10

REFERENCES

- I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless Sensor Networks: A Survey,” *Computer Networks*, vol. 38, no. 4, pp. 393-422, Mar. 2002.
- C. Vu, R. Beyah, and Y. Li, “A Composite Event Detection in Wireless Sensor Networks,” *Proc. IEEE Int’l Performance, Computing, and Comm. Conf. (IPCCC ’07)*, Apr. 2007.
- S. Uluagac, C. Lee, R. Beyah, and J. Copeland, “Designing Secure Protocols for Wireless Sensor Networks,” *Wireless Algorithms, Systems, and Applications*, vol. 5258, pp. 503-514, Springer, 2008.
- G.J. Pottie and W.J. Kaiser, “Wireless Integrated Network Sensors,” *Comm. ACM*, vol. 43, no. 5, pp. 51-58, 2000.