**COMPUTER NETWORKS**

**CSE-323**

**PROJECT-SECRETED**

Server Enabled and Client Riveted chat Embedding TCP sockets and Encode-Decoding

SUBMITTED TO:

PROF. SIVANESAN S

SUBMITTED BY:

AYUSHI NIGAM – 12BCE0410

SAURABH SINHA – 12BCE0598

CHAITANYA DHINGRA – 12BCE0095

SLOT: E2

**Contents:**

1. **Abstract**
2. **Introduction**
3. **In context with the IEEE paper**
4. **Existing work**
5. **Proposed System**
6. **Conceptual Diagram**
7. **Implementation**
8. **Testing and analysis**
9. **Conclusions and Future Enhancements**
10. **References**

**ABSTRACT**

In olden times, communication use to happen through letters, it took days and sometimes months to reach the destination. Moreover, security of the message was a serious issue since the letter could be read by anybody and this lead to leakage of confidential information. In those days, people used ciphers to convey important and confidential information. With the advancing technology there has been a rapid change in the field of communication technology. Firstly came the era of computer where machines were designed to solve computational problems and development of algorithm for solving various mathematical problems. Low level languages were used for feeding in the algorithm and inputs in the computing machine called computers. And since then there has been continuous progress and development in the field of computers.

In this journey of computer evolution there have been many stages and then one of the most significant steps was the merging of computers and communications which had a profound influence on the way computer systems were organized. It has completely changed the meaning of the word communication. The concept of the ''computer center'' as a room with a large computer to which users bring their work for processing is now totally obsolete. The old model of a single computer serving all of the organization's computational needs has been replaced by one in which a large number of separate but interconnected computers do the job. These combined computational systems are called computer networks. Two computers are said to be interconnected if they are able to exchange information. Networks come in many sizes, shapes and forms. These networks have completely changed the definition of communication. In today’s world, one person can send a message from one corner of the earth to the other in fraction of seconds. One can keep in touch with all their relatives and friends at all point of time whatever be the distance between them. All this is possible today because of computer networking. The wide spread network connects all of us together.

The developing network communication technology has brought a lot of changes in the modern society. Today, people send e-mail to each other for general communication. Moreover, LAN networks in offices made it much easier and efficient for the employees to communicate and co-ordinate with each other. Also business has started growing on networks rapidly, where sellers can directly sell their gods to the target customers without even having a retail shop and also the buyers get the advantage of purchasing the goods according to their desires and their access to goods is not dependent on the availability in the local market. Thus, it has promoted global marketing. Instant messaging is also a key feature of networking. These are chat platforms where duplex communication takes place between two people over a network. There are various applications available in the market for the same. They are being given the name of social networking sites as on these platforms people can socialise and interact with each other and form groups and circles.

One essential aspect for secure communications is that of Cryptography. Cryptography is the art of achieving security by encoding messages to make them non-readable. The concept of securing messages through cryptography has a long history. Indeed, Julius Caesar is credited with creating one of the earliest cryptographic systems to send military messages to his generals. Cryptography is the science of using mathematics to encrypt and decrypt data. Cryptography enables you to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient. While cryptography is the science of securing data, cryptanalysis is the science of analyzing and breaking secure communication.

In our project, we have combined the two key features of networking which are communication and network security. Firstly we have done a detailed a detailed study of the various encryption and decryption algorithms, where encryption is a process of coding information which could either be a file or mail message into a cipher text a form unreadable without a decoding key in order to prevent anyone except the intended recipient from reading the data. Decryption is the reverse process of converting encoded data to its original un-encoded form, plaintext. In this project, we have devised our algorithm for encryption and decryption. The proposed algorithm is a symmetric encryption algorithm. Symmetric encryption is the most common type of encryption and uses the same key for encoding and decoding data. This key is known as a session key.

The above mentioned algorithm is then being implemented in a chat application. We have put forward the design method of network chat system based on Socket and have also implemented the same using the symmetric encryption algorithm. Socket interface is an application programming interface of TCP/IP network, this interface definition of a number of functions or routine, programmers can use them to the development of TCP / IP network applications. Java Socket is tailored for the Java language by SUN which is used to develop network application program. We have further discussed about the Socket interface and have also laid emphasis oncloud computing technology as an important concept of, analyze the Socket communication mechanism in principle, of the application of Java Socket. In order to improve the communication network chat in the data stored safety and efficiency, specifically in the Socket technology application based add on cloud computing technology for the improvement of the method has been implemented with a newly devised encryption algorithm.

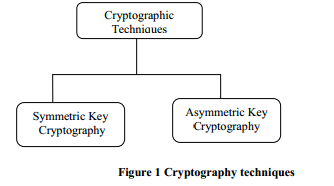
The system being implemented is a server client chat system with a highly secure connection making use of the algorithm devised in the beginning. This enhances the security of transmission of data over a network. We have named the newly devised algorithm as ‘Secret’ which is a multi-purpose encryption algorithm and can be implemented on a variety of platforms. In our project the execution of ‘Secret’ is being displayed with the help of socket programming chat application.

**INTRODUCTION**

During this time when the Internet provides essential communication between tens of millions of people and is being increasingly used as a tool for commerce, security becomes a tremendously important issue to deal with. There are many aspects to security and many applications, ranging from secure commerce and payments to private communications and protecting Passwords. There are many aspects to security and many applications, ranging from secure commerce and payments to private communications and protecting passwords decryption. A. In a typical situation where cryptography is used, two parties (X and Y) communicate over an insecure channel. X and Y want to ensure that their communication remains incomprehensible by anyone who might be listening. Furthermore, because X and Y are in remote locations, X must be sure that the information she receives from Y has not been modified by anyone during transmission. In addition, she must be sure that the information really does originate from Y and not someone impersonating Y.

Encryption algorithm

Encryption algorithm or cipher is a mathematical function used in the encryption and decryption process - series of steps that mathematically transforms plaintext or other readable information into unintelligible cipher text. A cryptographic algorithm works in combination with a key (a number, word, or phrase) to encrypt and decrypt data. To encrypt, the algorithm mathematically combines the information to be protected with a supplied key. The result of this combination is the encrypted data. To decrypt, the algorithm performs a calculation combining the encrypted data with a supplied key. The result of this combination is the decrypted data. If either the key or the data is modified, the algorithm produces a different result. The goal of every encryption algorithm is to make it as difficult as possible to decrypt the generated cipher text without using the key. Some cryptographic methods rely on the secrecy of the encryption algorithms; such algorithms are only of historical interest and are not adequate for real-world needs. Instead of the secrecy of the method itself, all modern algorithms base their security on the usage of a key; a message can be decrypted only if the key used for decryption matches the key used for encryption.



There are two kinds of key-based encryption algorithms, symmetric encryption algorithms also called as secret key algorithms and asymmetric encryption algorithms also referred to as public key algorithms as shown in figure1. The difference is that symmetric encryption algorithms use the same key for encryption and decryption (or the decryption key is easily derived from the encryption key), whereas asymmetric encryption algorithms use a different key for encryption and decryption, and the decryption key cannot be derived from the encryption key. In secret key cryptography, a single key is used for both encryption and decryption. the sender uses the key (or some set of rules) to encrypt the plaintext and sends the cipher text to the receiver. The receiver applies the same key to decrypt the message and recover the plaintext. Because a single key is used for both functions, secret key cryptography is also called symmetric encryption. With this form of cryptography, it is obvious that the key must be known to both the sender and the receiver. The biggest difficulty with this approach, of course, is the distribution of the key.

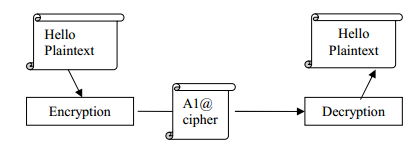


Figure 2: Symmetric key cryptography

The most widely used symmetric key cryptographic method is the Data Encryption Standard (DES) , published in 1977 by the National Bureau of Standards.  DES It is still the most widely used symmetric-key approach. It uses a fixed length, 56-bit key and an efficient algorithm to quickly encrypt and decrypt messages. It can be easily implemented in hardware, making the encryption and decryption process even faster. In general, increasing the key size makes the system more secure. A variation of DES, called Triple-DES or DES-EDE (encrypt-decrypt-encrypt), uses three applications of DES and two independent DES keys to produce an effective key length of 168 bits [ANSI 85].

The International Data Encryption Algorithm (IDEA) was invented by James Massey and Xuejia Lai of ETH Zurich, Switzerland in 1991. IDEA uses a fixed length, 128-bit key (larger than DES but smaller than Triple-DES). It is also faster than Triple-DES. In the early 1990s, Don Rivest of RSA Data Security, Inc., invented the algorithms RC2 and RC4. These use variable length keys and are claimed to be even faster than IDEA. However, implementations may be exported from the U.S. only if they use key lengths of 40 bits or fewer.

Discussing of asymmetric key cryptography, it overcomes the key management problem by using different encryption and decryption key pairs. Having knowledge of one key, say the encryption key, is not sufficient enough to determine the other key - the decryption key. Therefore, the encryption key can be made public, provided the decryption key is held only by the party wishing to receive encrypted messages (hence the name public/private key cryptography). Anyone can use the public key to encrypt a message, but only the recipient can decrypt it.

RSA  is a widely used public/private key algorithm is, named after the initials of its inventors, Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman [RSA 91]. It depends on the difficulty of factoring the product of two very large prime numbers. Although used for encrypting whole messages, RSA is much less efficient than symmetric key algorithms such as DES. ElGamal is another public/private key algorithm [El Gamal 85]. This uses a different arithmetic algorithm than RSA, called the discrete logarithm problem.The mathematical relationship between the public/private key pair permits a general rule: any message encrypted with one key of the pair can be successfully decrypted only with that key's counterpart. To encrypt with the public key means you can decrypt only with the private key. The converse is also true - to encrypt with the private key means you can decrypt only with the public key.

In the above section we have discussed in detail about cryptography, a domain of network security. Now, we need to have a deeper insight on the terms ‘network’ and ‘communication’. In today's network times, "Network" this word already thorough popular feeling, the development of computer. Network speed is amazing, it greatly reduces the human distance, and it apparently expanded the computer functions.

As defines in Wikipedia “A computer network or data network is a [telecommunications network](http://en.wikipedia.org/wiki/Telecommunications_network) that allows [computers](http://en.wikipedia.org/wiki/Computer) to exchange [data](http://en.wikipedia.org/wiki/Data_(computing)). In computer networks, networked computing devices pass data to each other along data connections. Data is transferred in the form of packets. The connections ([network links](http://en.wikipedia.org/wiki/Data_link)) between nodes are established using either [cable media](http://en.wikipedia.org/wiki/Transmission_line) or [wireless media](http://en.wikipedia.org/wiki/Wireless_network). The best-known computer network is the [Internet](http://en.wikipedia.org/wiki/Internet)”.

At a basic level, network-based systems consist of a server, client, and a media for communication as shown in Fig 2. A computer running a program that makes a request for services is called client machine. A computer running a program that offers requested services from one or more clients is called server machine. The media for communication can be wired or wireless network. Generally, programs running on client machines make requests to a program (often called as server program) running on a server machine. They involve networking services provided by the transport layer, which is part of the Internet software stack, often called TCP/IP (Transport Control Protocol/Internet Protocol) stack. The transport layer comprises two types of protocols, TCP (Transport Control Protocol) and UDP (User Datagram Protocol). The most widely used programming interfaces for these protocols are sockets.

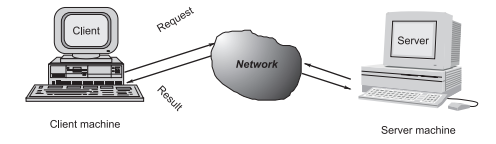


Figure 3: Client-Server Communication

One of the key networking application is chatting tools, it is also more and more attention by people and welcome, a good network chat tool can make any distance communication convenient and quick. In fact, in modern social network chat software development has quite mature, all kinds of complete function and friendly interface of software is quite beautiful. There into, based on the Socket of the chat system is such communication software one of the specific examples, it is good to interpret the Socket communication principle and it in the enterprise internal communication, teaching, and discussed the application of certain practical value. It is based on TCP connection between two processes of end-to-end communication mechanism. It has to send and receive information fast speed, good secrecy and take up the network bandwidth resources is low, take up the server low throughput capacity, easy to the programming.

Sockets provide an interface for programming networks at the transport layer. Network communication using Sockets is very much similar to performing ﬁ le I/O. In fact, socket handle is treated like ﬁ le handle. The streams used in ﬁle I/O operation are also applicable to socket-based I/O. Socket-based communication is independent of a programming language used for implementing it. That means, socket program written in Java language can communicate to a program written in non-Java (say C or C++) socket program. A server (program) runs on a speciﬁc computer and has a socket that is bound to a specific port. The server listens to the socket for a client to make a connection request (Fig 2(a)). If everything goes

well, the server accepts the connection (Fig 2(b)). Upon acceptance, the server gets a new socket bound to a different port. It needs a new socket (consequently a different port number) so that it can continue to listen to the original socket for connection requests while serving the connected client. The java.net.Socket class represents a socket, and the java.net.ServerSocket class provides a mechanism for the server program to listen for clients and establish connections with them.

The java.net package of the J2SE APIs contains a collection of classes and interfaces that provide the low-level communication details, allowing you to write programs that focus on solving the problem at hand.

The java.net package provides support for the two common network protocols:

* **TCP:** TCP stands for Transmission Control Protocol, which allows for reliable communication between two applications. TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.
* **UDP:** UDP stands for User Datagram Protocol, a connection-less protocol that allows for packets of data to be transmitted between applications.

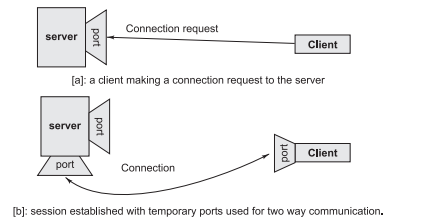


Figure 4

This project makes use of a symmetric key algorithm which has been implemented in a chat application. The chat application supports two way communications between the server and client. The chat application has been designed using socket programming in TCP/IP domain. The programming language used is JAVA. Java provides a set of classes, deﬁned in a package called java.net, to enable the rapid development of network applications. It also provides key classes, interfaces, and exceptions in java.net package simplifying the complexity involved in creating client and server programs. We a have aimed at designing highly secure chat systems for very confidential conversations where leakage of any information can be a matter of heavy consequences.

Moreover, we have done theoretical analysis of the usage of could technology in chat application and have formulated a design which shows that implementation of cloud technology enhances network communication. Cloud computing is distributed processing, parallel processing and the development of grid computing, or is it the

computer science concept of commercial realized.

Cloud calculative fundamental is, through the analysis of the distribution of the distributed computer, rather than the local computer or the remote server, the operation of the enterprise data center will be more similar as Internet. This makes the enterprise can will resource switch on the application of need, according to demand access to the computer and storage system. This is a kind of revolutionary

Act, for example, this is just like is the old single generator model turned to the power plant to centralized power supply mode. It means computational ability to also can serve as a kind of goods circulation, as gas, water and electricity, take with convenient , fare is low. The biggest different depend on, it is through the Internet for transmission. Cloud computing has the blueprint of be vividly portrayed: in the future, only need a laptop or a mobile phone, you can realize our need through network service everything, even including supercomputing such task. From this perspective, the end user is the cloud calculative true owner. Cloud computing applications include such a new idea or new concept: the power of the world together, to give which every member of the using, as shown in Figure 3 shows.

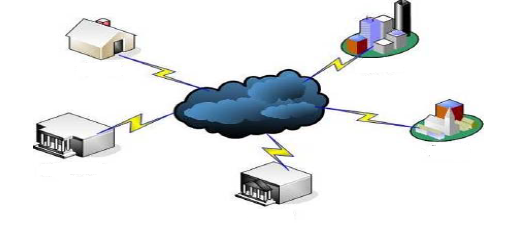


Figure 5

**In context with the IEEE paper**

The IEEE paper being referred has the title, “*The Design Method of Network Chat System Based on Socket and Cloud Computing* “. The paper lays emphasis on a socket chat application based on cloud computing. The paper has firstly described cloud computing in details. They have defined the usage of cloud computing as the new developing tehnology which has the power of bringing about revolution in the field of computer science. "Cloud" benefits are that one of the computers can update, guarantee "cloud" forever. Google is byseveral such "clouds" of composition, other IT giants such as Microsoft, Yahoo and Amazon also have or are building this"cloud". At the time, we just need a computer can connect to the Internet, do not need to care about storage or computing happen where a "cloud", but once a need, we can be in any place with any equipment, such as computers, mobile phones,fast calculation and find these material. We will no longerhave to worry about material lost. It uses a phase in reference to cloud computing “*The World Only Need 5 Computers”.* According to the phrase, cloud computing is considered as the science and technology industry of the next revolution, it will bring workmethods and business model of fundamental change. First of all, to small and medium enterprises and entrepreneurs for,cloud computing means huge commercial opportunity, theycan use cloud computing at a higher level and large enterprisecompetition.

The paper chat system proposed in the paper is based on Socket and can be regarded as two program in one of the communication link endpoint, a program will write a message in the Socket, the Socket will this period of information sent to another in the Socket, it makes this period of information can be transmitted to other applications, its communication. The system put completing interface design of system to use the component element AWT and Swing in Java, it build communication in both ends based on Socket port, introducing multithreading technology realize concurrent execution, using JDBC database connection technology to create a database connection.

The various functions implemented in the chat are: (a)Function of Users Login: Through the fill in the user name and password, choose the right to enter chat room on the interface. (b) Function of User Registration: Fill in and submit to a server in the personal information, server registered information will become successful after members. Submit personal information, the client to personal information of validation.(c)Group Chat: user can choose to interface in group chat（chat room the interface） online of all users or specific users send information, and receive other customer information. Private Chat: The user can

chat room facing the world in a particular user sending private chat request, the other party has accepted, both sides will enter private chat interface to private chat.

(d)Function of Warning and Kicking: The client and server administrator managers are warning and kicking through the rights to manage chat rooms.

This paper introduces a Socket communication principle and the application in JAVA, and on this basis, to a network chat system development process was described. The system is based on Socket and cloud computing concept realized by the point-to-point chat system, through the Java Swing components provide the powerful graphics interface editing functions, create a user interface of beautiful and friendly, and combined with Java multithreading technology realize the client chat messages of sending and receiving, many users group chat and point to some private chat the concurrent execution, etc. Cloud computing is a kind of Internet super calculation mode, in a remote data center, hundreds of thousands of computers and servers connected to a computing cloud. Therefore, cloud computing can even let you experience ten trillion times per second of the operation ability, have so powerful computing power can simulate a nuclear blast, predict climate change, market development trend and chat online data storage. No matter where the world the user can through the computer, mobile phone and cloud computing access the way data center, you can be more efficient, safe to chat and the Internet,according to the needs of their proceed computing.

The other paper referred is “A Symmetric Key Cryptographic Algorithm” which was published in ©2010 International Journal of Computer Applications (0975 - 8887)

Volume 1 – No. 15. The abstract of the paper defines cryptography and its types which exaplined in detail in the futhur in the paper.The paper describes cryptography as the art of achieving security by encoding messages to make them non-readable. Cryptography is the practice and study of hiding information. In modern times

cryptography is considered a branch of both mathematics and computer science and is affiliated closely with information theory, computer security and engineering. Cryptography is used in applications present in technologically advanced societies;

examples include the security of ATM cards, computer passwords and electronic commerce, which all depend on cryptography.

A cryptographic algorithm, or cipher, is a mathematical function used in the encryption and decryption process. A cryptographic algorithm works in combination with a key—a word, number, or phrase—to encrypt the plaintext. The same plaintext encrypts to different ciphertext with different keys. The security of encrypted data is entirely dependent on two things: the strength of the cryptographic algorithm and the secrecy of the key. A cryptographic algorithm, plus all possible keys and all the protocols that make it work comprise a cryptosystem. It further how cryptography is used to achieve the following goals:

* Confidentiality

To ensure data remains private. Confidentiality is usually achieved using encryption. Encryption algorithms (that use encryption keys) are used to convert plain text into cipher text and the equivalent decryption algorithm is used to convert the cipher text back to plain text. Symmetric encryption algorithms use the same key for encryption and decryption, while asymmetric algorithms use a public/private key pair.

* Data integrity

To ensure data is protected from accidental or deliberate (malicious) modification. Integrity is usually provided by message authentication code or hashes. A hash value is a fixed length numeric value derived from a sequence of data. Hash values are used to verify the integrity of data sent through insecure channels. The hash value of received data is compared to the hash value of the data as it was sent to determine if the data was altered.

* Authentication

To assure that data originates from a particular party.Digital certificates are used to provide authentication. Digital signatures are usually applied to hash values as these are significantly smaller than the source data that they represent.

Symmetric key cryptography

Secret key cryptography schemes are generally categorized as being either stream ciphers or block ciphers. Stream ciphers operate on a single bit (byte or computer word) at a time, and implement some form of feedback mechanism so that the key is constantly changing. A block cipher is so-called because the scheme encrypts one block of data at a time using the same key on each block. In general, the same plaintext block will always encrypt to the same ciphertext when using the same key in a block cipher whereas the same plaintext will encrypt to different ciphertext in a stream cipher .Stream ciphers come in several flavors but two are worth mentioning here. Self-synchronizing stream ciphers calculate each bit in the keystream as a function of the previous n bits in the keystream. It is termed "self-synchronizing" because the decryption process can stay synchronized with the encryption process merely by knowing how far into the n-bit keystream it is. Synchronous stream ciphers generate the keystream in a fashion independent of the message stream but by using the same keystream generation function at sender and receiver. While stream ciphers do not propagate transmission errors, they are, by their nature, periodic so that the keystream will eventually repeat.

Block ciphers can operate in one of several modes; the following four are the most important: Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB) mode and Output Feedback (OFB). The most common secret-key cryptography scheme used today is the Data Encryption Standard (DES), designed by IBM in the 1970s and adopted by the National Bureau of Standards (NBS) [now the National Institute for Standards and Technology (NIST)] in 1977 for commercial and unclassified government applications. DES has been adopted as Federal

Information Processing Standard 46 (FIPS 46- 3) and by the American National Standards Institute as X3.92).

The new symmetric algorithm proposed in the paper was:

Encryption algorithm

Step 1: Generate the ASCII value of the letter

Step 2: Generate the corresponding binary value of it.

[Binary value should be 8 digits e.g. for decimal 32 binary number

should be 00100000]

Step 3: Reverse the 8 digit’s binary number

Step 4: Take a 4 digits divisor (>=1000) as the Key

Step 5: Divide the reversed number with the divisor

Step 6: Store the remainder in first 3 digits & quotient in next 5 digits (remainder and quotient wouldn’t be more than 3 digits and 5 digits long respectively. If any of these are less then 3 and 5 digits respectively we need to add required number of 0s (zeros) in the left hand side. So, this would be the cipertext i.e. encrypted text. Now store the remainder in first 3 digits & quotient in next 5 digits.

Implementation:

Let, the character is “T”. Now according to the steps we will get the following:

Step 1: ASCII of “T” is 84 in decimal.

Step 2: The Binary value of 84 is 1010100. Since it is not an 8 bit binary number we need to make it 8 bit number as per the encryption algorithm. So it would be 01010100

0 1 0 1 0 1 0 0

Step 3: Reverse of this binary number would be 00101010

0 0 1 0 1 0 1 0

Step 4: Let 1000 as divisor i.e. Key

Step 5: Divide 00101010 (dividend) by 1000(divisor)

Step 6: The remainder would be 10 and the quotient would be 101.

So as per the algorithm the ciphertext would be 01000101 which is

ASCII 69 in decimal i.e. “E”

0 1 0 0 0 1 0 1

Decryption algorithm

Step 1: Multiply last 5 digits of the ciphertext by the Key

Step 2: Add first 3 digits of the ciphertext with the result produced

in the previous step

Step 3: If the result produced in the previous step i.e. step 2 is not

an 8-bit number we need to make it an 8- bit number

Step 4: Reverse the number to get the original text i.e. the plain

Text.

Thus, the paper being used as reference in the project has been described in detail above. They throw light on socket programming and developing a chat application using the same. Not only that it also gives an insight into cloud technology and describes how cloud technology usage in chat system can increase the efficiency. The other paper referred is a non-IEEE paper. It discusses about symmetric key encryption. The paper explains cryptography and its types like symmetric key encryption and asymmetric key encryption in detail. In the latter section it decribes about symmetric key. It also proposes a new algorithm for symmetric key encoding.

**EXISTING WORK**

The base of the application using socket programming is the Chat System implemented by uncountable peers around the industry. It uses Ports to connect to the Device and further establishes the connection with the Application through the Socket linking the two applications. The Server is first established and waits for a socket to be connected to it for a limited listed period of the time after which the open portal socket expires.   
  
There are various versions of the socket programmed application incorporating varying communication links namely Simplex, Half-Duplex, Full-Duplex Communication.

The Simplex communication is one-sided, generally from server to client and generally used in applications for information distribution where the server is pinged by the client and the server returns a pre defined set of information such as the Weather Report, Status of a Process or Various similar purposes which doesn’t require any communicative feedback from the client.

Then, we have the half-duplex link establishing a two-sided communication where only one-way communication is possible at a particular given instance. The communication is established with a request and data packet. The reciever on receiving the data acknowledges the received packet and then process the packet.

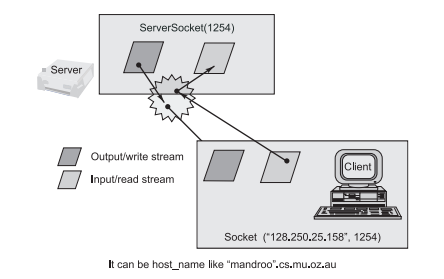


Figure 6:Socket based client-server communication

Using socket programming for developing network applications is made possible in Java by using sockets, threads, RMI, clustering, and Web services. These technologies allow for the creation of portable, eﬃcient, and maintainable large and complex Internet applications. The java.net package provides a powerful

and ﬂexible set of classes for implementing network applications. Typically, programs running on client machines make requests to programs on a server

Machine. These involve networking services provided by the transport layer.

The most widely used transport protocols on the Internet are TCP (Transmission control Protocol) and UDP (User Datagram Protocol). TCP is a connection-oriented protocol providing a reliable ﬂow of data between two computers. It is used by applications such as the World Wide Web, e-mail, p, and secure shell. On the other hand, UDP is a simpler message-based connectionless protocol which sends packets of data known as datagrams from one computer to another with no guarantees of arrival. Network applications using UDP include Domain Name Systems (DNS),streaming media applications as IPTV, VoIP, and online games. Sockets provide an interface for programming networks at the transport layer. Using sockets, network communication is very much similar to performing ﬁ le I/O. A socket is an endpoint of a two-way communication link between two programs running on the network. The source and destination IP address, and the port numbers constitute a network socket. Two key classes from java.net package used in creation of server and client programs are ServerSocket, which represents a server socket, and Socket, an instantiation of which performs the actual communication with the client. Datagram communication is also supported through Datagram Packet and Datagram Socket classes. Writing and reading data between server and the client is also supported through the URLconnection class.

URL Encoding

It is very important that the Web can be accessed via heterogeneous platforms such as Windows, Linux, or Mac. The characters in the URL must come from a ﬁ xed subset of ASCII in order to maintain the interoperability between various platforms. Speciﬁcally, the capital letters A–Z, the lowercase letters a–z, the digits 0–9 and the punctuation characters. Encoding is very simple, any characters that are not ASCII

numerals, letters, or the punctuation marks allowed are converted into bytes and each byte is written as a percentage sign followed by two hexadecimal digits. For example, the following program helps encode a query string if non-ASCII characters are present.

Some of the popular symmetric key encryption algorithms are:

**AES encryption algorithm**

AES stands for Advanced Encryption Standard. AES is a [symmetric key encryption](http://www.encryptionanddecryption.com/encrypt_decrypt_encyclopedia.html#Symmetric_Key) technique which will replace the commonly used [Data Encryption Standard (DES)](http://www.encryptionanddecryption.com/algorithms/encryption_algorithms.html#DES). It was the result of a worldwide call for submissions of encryption algorithms issued by the US Government's National Institute of Standards and Technology (NIST) in 1997 and completed in 2000.  
In response to the growing feasibility of attacks against DES, NIST launched a call for proposals for an official successor that meets 21st century security needs. This successor is called the Advanced Encryption Standard (AES).  
Five algorithms were selected into the second round, from which Rijndael was selected to be the final standard. NIST gave as its reasons for selecting Rijndael that it performs very well in hardware and software across a wide range of environments in all possible modes. It has excellent key setup time and has low memory requirements, in addition its operations are easy to defend against power and timing attacks. NIST stated that all five finalists had adequate security and that there was nothing wrong with the other four ciphers.   
The winning algorithm, Rijndael, was developed by two Belgian cryptologists, Vincent Rijmen and Joan Daemen.  
AES provides strong encryption and was selected by NIST as a Federal Information Processing Standard in November 2001 (FIPS-197).  
Rijndael follows the tradition of square ciphers. AES algorithm uses three key sizes: a 128-, 192-, or 256-bit encryption key. Each encryption key size causes the algorithm to behave slightly differently, so the increasing key sizes not only offer a larger number of bits with which you can scramble the data, but also increase the complexity of the cipher algorithm.

**Blowfish encryption algorithm**

Blowfish is a [symmetric encryption algorithm](http://www.encryptionanddecryption.com/encrypt_decrypt_encyclopedia.html#Symmetric_Encryption) designed in 1993 by Bruce Schneier as an alternative to existing encryption algorithms.  
Blowfish has a 64-bit block size and a variable key length - from 32 bits to 448 bits. It is a 16-round [Feistel cipher](http://www.encryptionanddecryption.com/encrypt_decrypt_encyclopedia.html#Feistel_Cipher) and uses large key-dependent [S-boxes](http://www.encryptionanddecryption.com/encrypt_decrypt_encyclopedia.html#Substitution_Box). While doing key scheduling, it generates large pseudo-random lookup tables by doing several encryptions. The tables depend on the user supplied key in a very complex way. This approach has been proven to be highly resistant against many attacks such as differential and linear cryptanalysis. Unfortunately, this also means that it is not the algorithm of choice for environments where a large memory space is not available. Blowfish is similar in structure to [CAST-128](http://www.encryptionanddecryption.com/encrypt_decrypt_encyclopedia.html#CAST), which uses fixed S-boxes.  
  
Since then Blowfish has been analyzed considerably, and is gaining acceptance as a strong encryption algorithm.  
Blowfish was designed in 1993 by Bruce Schneier as a fast, free alternative to existing encryption algorithms. Since then it has been analyzed considerably, and it is slowly gaining acceptance as a strong encryption algorithm. Blowfish is unpatented and license-free, and is available free for all uses.  
  
The only known attacks against Blowfish are based on its weak key classes.

**CAST**

CAST stands for Carlisle Adams and Stafford Tavares, the inventors of CAST. CAST is a popular 64-bit block cipher which belongs to the class of encryption algorithms known as Feistel ciphers.  
CAST-128 is a DES-like Substitution-Permutation Network (SPN) cryptosystem. It has the Feistel structure and utilizes eight fixed S-boxes. CAST-128 supports variable key lenghts between 40 and 128 bits.   
CAST-128 is resistant to both linear and differential cryptanalysis. Currently, there is no known way of breaking CAST short of brute force. CAST is now the default cipher in PGP.

**Data Encryption Standard (DES)**

Digital Encryption Standard (DES) is a symmetric [block cipher](http://www.encryptionanddecryption.com/encrypt_decrypt_encyclopedia.html#Block_Cipher) with 64-bit block size that uses using a 56-bit key.In 1977 the Data Encryption Standard (DES), a [symmetric algorithm](http://www.encryptionanddecryption.com/encrypt_decrypt_encyclopedia.html#Symmetric_Algorithms), was adopted in the United States as a federal standard.  
  
DES encrypts and decrypts data in 64-bit blocks, using a 56-bit key. It takes a 64-bit block of plaintext as input and outputs a 64-bit block of ciphertext. Since it always operates on blocks of equal size and it uses both permutations and substitutions in the algorithm. DES has 16 rounds, meaning the main algorithm is repeated 16 times to produce the ciphertext. It has been found that the number of rounds is exponentially proportional to the amount of time required to find a key using a brute-force attack. So as the number of rounds increases, the security of the algorithm increases exponentially.  
  
For many years, DES-enciphered data were safe because few organizations possessed the computing power to crack it. But in July 1998 a team of cryptographers cracked a DES-enciphered message in 3 days, and in 1999 a network of 10,000 desktop PCs cracked a DES-enciphered message in less than a day. DES was clearly no longer invulnerable and since then [Triple DES (3DES)](http://www.encryptionanddecryption.com/encrypt_decrypt_encyclopedia.html#Triple_DES) has emerged as a stronger method.  
  
Triple DES encrypts data three times and uses a different key for at least one of the three passes giving it a cumulative key size of 112-168 bits. That should produce an expected strength of something like 112 bits, which is more than enough to defeat brute force attacks. Triple DES is much stronger than (single) DES, however, it is rather slow compared to some new block ciphers. However, cryptographers have determined that triple DES is unsatisfactory as a long-term solution, and in 1997, the National Institute of Standards and Technology (NIST) solicited proposals for a cipher to replace DES entirely, the [Advanced Encryption Standard (AES)](http://www.encryptionanddecryption.com/encrypt_decrypt_encyclopedia.html#AES).

**IDEA encryption algorithm**

IDEA stands for International Data Encryption Algorithm. IDEA is a symmetric encryption algorithm that was developed by Dr. X. Lai and Prof. J. Massey to replace the DES standard. Unlike DES though it uses a 128 bit key. This key length makes it impossible to break by simply trying every key. It has been one of the best publicly known algorithms for some time. It has been around now for several years, and no practical attacks on it have been published despite of numerous attempts to analyze it.  
IDEA is resistant to both linear and differential analysis.

**RC2**

RC2 is a variable-key-length cipher. It was invented by Ron Rivest for RSA Data Security, Inc. Its details have not been published.

**RC4**

RC4 was developed by Ron Rivest in 1987. It is a variable-key-size stream cipher. It is a cipher with a key size of up to 2048 bits (256 bytes). The algorithm is very fast. Its security is unknown, but breaking it does not seem trivial either. Because of its speed, it may have uses in certain applications. It accepts keys of arbitrary length. RC4 is essentially a pseudo random number generator, and the output of the generator is exclusive-ored with the data stream. For this reason, it is very important that the same RC4 key never be used to encrypt two different data streams.

**RC6**

RC6 is a symmetric key block cipher derived from RC5. It was designed by Ron Rivest, Matt Robshaw, Ray Sidney, and Yiqun Lisa Yin to meet the requirements of the Advanced Encryption Standard (AES) competition. RC6 encryption algorithm was selected among the other finalists to become the new federal Advanced Encryption Standard (AES).

**SEED**

SEED is a block cipher developed by the Korea Information Security Agency since 1998. Both the block and key size of SEED are 128 bits and it has a Feistel Network structure which is iterated 16 times. It has been designed to resist differential and linear cryptanalysis as well as related key attacks. SEED uses two 8x8 S-boxes and mixes the XOR operation with modular addition. SEED has been adopted as an ISO/IEC standard (ISO/IEC 18033-3), an IETF RFC, RFC 4269 as well as an industrial association standard of Korea (TTAS.KO-12.0004/0025).

**Serpent**

Serpent is a very fast and reasonably secure block cipher developed by Ross Anderson, Eli Biham and Lars Knudsen. Serpent can work with different combinations of key lengths. Serpent was also selected among other five finalists to become the new federal Advanced Encryption Standard (AES).

**TEA**

Tiny Encryption Algorithm is a very fast and moderately secure cipher produced by David Wheeler and Roger Needham of Cambridge Computer Laboratory. There is a known weakness in the key schedule, so it is not recommended if utmost security is required. TEA is provided in 16 and 32 round versions. The more rounds (iterations), the more secure, but slower.

**Triple DES**

Triple DES is a variation of Data Encryption Standard (DES). It uses a 64-bit key consisting of 56 effective key bits and 8 parity bits. The size of the block for Triple-DES is 8 bytes. Triple-DES encrypts the data in 8-byte chunks. The idea behind Triple DES is to improve the security of DES by applying DES encryption three times using three different keys. Triple DES algorithm is very secure (major banks use it to protect valuable transactions), but it is also very slow.

**Twofish**

Twofish is a symmetric block cipher. Twofish has a block size of 128 bits and accepts keys of any length up to 256 bits. Twofish has key dependent S-boxes like Blowfish. Twofish encryption algorithm was designed by Bruce Schneier, John Kelsey, Chris Hall, Niels Ferguson, David Wagner and Doug Whiting. The National Institute of Standards and Technology (NIST) investigated Twofish as one of the candidates for the replacement of the DES encryption algorithm.

The algorithm which is used as a base algorithm for developing the new symmetric key algorithm ‘secret’ is Caesar’s cipher. **Caesar shift**, is one of the simplest and most widely known[encryption](http://www.wikipedia.org/wiki/Encryption) techniques. It is a type of [substitution cipher](http://www.wikipedia.org/wiki/Substitution_cipher) in which each letter in the[plaintext](http://www.wikipedia.org/wiki/Plaintext) is replaced by a letter some fixed number of positions down the [alphabet](http://www.wikipedia.org/wiki/Alphabet). For example, with a shift of 3, A would be replaced by D, B would become E, and so on. The method is named after [Julius Caesar](http://www.wikipedia.org/wiki/Julius_Caesar), who used it to communicate with his generals.

**Proposed System**

In the project we have designed a symmetric key encryption algorithm. Taking caesar cipher as the based algorithm where in the encrypted string each character is shifted by a constant number. through this algorithm we have aimed at developing a highly secure encryption method.

The key characteristics of the proposed algorithm:

1.We have used the principle of caesar cipher , have added the number 1 and subtracted the number 1 with the ASCII value of the characters at alternate position.

2. We have came up the new idea of padding a number. No popular algorithm uses this concept, thus the new concept of padding in encryption have been developed by us. In this concept we introduce a random number padding at the LSB and MSB of each of the character.

3.Another important concept implemented is division by a large prime number which is the key. Thus, each character divides the key and the quotient which is in decimal is being sent to the receiver.

The proposed algorithm has been illustrated in the form of flow chart. Figure 7 displays the steps at the sender’s side where data is being encrypted while figure 8 displays the steps at receiver's side where data is being decrypted. From the flow charts we can see that the steps used in on the receiver's side are reverse of that of the sender’s side and the complementary mathematical functions are being used at the receiver’s side. For decrypting the message each character of the message have to be multiplied by the same key used to encrypt the data. Also one important point is that after removing the padding alternate characters have to be subtracted and added to 1. We need to take care of the order of subtracting at both the side’s , receivers and sender’s.

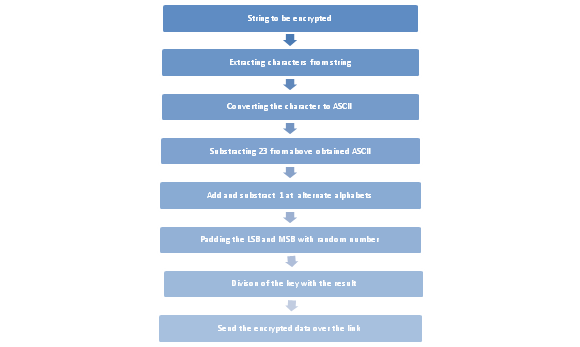


                                       Figure 7: Encryption



      Figure 8:Decryption

Key factor:

From the above explanation we see that the most important part of the algorithm is the key being used. The key should be a very large prime number. We prefer the use of a very large prime number because it will be difficult to hack such a system. The users of the algorithm have to name of the algorithm is secret since the algorithm has to be kept secret. Only the sender and receiver of the algorithm would know the key and step of the algorithm. This kind of system can be used for highly secure communication which is related to national security or even in personal conversations where one cannot risk getting the information leaked. The padding characteristic of the algorithm cannot be easy known by the hack till he knows the exact steps of hacking.

In our project, we have used the above proposed algorithm, ‘Secret, in half duplex chat application. We have used socket programming for the intercommunication between programs running on different computers in the network. We have used TCP/IP socket programming for developing the chat system. A socket is an endpoint of a two-way communication link between two programs running on the network.

Socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent. Java provides a set of classes, deﬁ ned in a package called java.net, to enable the rapid development of network applications. Key classes, interfaces, and exceptions in java.net package simplifying the complexity involved in creating client and server programs are:

The Classes

* ContentHandler
* DatagramPacket
* DatagramSocket
* DatagramSocketImpl
* HttpURLConnection
* InetAddress
* MulticastSocket
* ServerSocket
* Socket
* SocketImpl
* URL
* URLConnection
* URLEncoder
* URLStreamHandler

The Interfaces

* ContentHandlerFactory
* FileNameMap
* SocketImplFactory
* URLStreamHandlerFactory350 Object-Oriented Programming with Java

Exceptions

* BindException
* ConnectException
* MalformedURLException
* NoRouteToHostException
* ProtocolException
* SocketException
* UnknownHostException
* UnknownServiceException

A simple Server Program in Java The steps for creating a simple server program are:

1. Open the Server Socket:

ServerSocket server = new ServerSocket( PORT );

2. Wait for the Client Request:

Socket client = server.accept(); Socket Programming 351

3. Create I/O streams for communicating to the client

DataInputStream is = new DataInputStream(client.getInputStream());

DataOutputStream os = new DataOutputStream(client.getOutputStream());

4. Perform communication with client

Receive from client: String line = is.readLine();

Send to client: os.writeBytes(“Hello\n”);

5. Close socket:

client.close();

Limitations and constraints

* The key have be kept confidential.
* Symbols having ASCII value less than 23 cannot be sent via this chat application.
* Half duplex communication.
* System has only one client.

Advantages of ‘Secret’:

* It has uses symmetric key.
* The usage of random numbers for padding makes it more secure.
* It is not feasible to hack into this system.
* The data being send is in float.

**Design and conceptual architecture**

Providing a secured communication service between two servers is what our project aims for. Secure communication is when two entities are communicating and do not want a third party to listen in. For that they need to communicate in a way not susceptible to eavesdropping or interception. Secure communication includes means by which people can share information with varying degrees of certainty that third parties cannot intercept what was said. Our team has researched few encryption and decryption algorithms used in various chat services and have created a unique algorithm which overcomes various security issues. Here is a conceptual sketch of what we are planning to do.

|  |
| --- |
| Computer 1:  String character of any length. |

|  |
| --- |
| Encrypted:  Encoded Data using our algorithm |

|  |
| --- |
| Data Transferred through IP address of the other client in Encrypted form. |

|  |
| --- |
| Computer 2: Decrypted:  Data is decoded and the String character is received. |

Figure 9

CONCRETE DIAGRAM:

The process of encryption in our algorithm takes place in steps to convert the character string data into a secured and anonymous form of representation which is only decoded using certain values from our algorithm in the receivers side. This way the data sent and received between the clients are secured and hidden from a third party. Based on various concepts of data encryption we have secured the data on various stages through coding and other operation implemented on it.

The data (character string) is sent from the sender computer to the receiver computer where the data is coded and operations are implemented on it in various stages :

Step 1:

The received character string is initially converted into an ASCII code (American Standard Code for Information Interchange) and 23 is subtracted from it which changes the characters into a two digit form. ASCII codes represent text in computer, communication equipment, and other devices that use text. Most modern character-encoding schemes are based on ASCII, though they support many additional characters.

Step 2:

The ASCII character is then incremented and decremented using a simple JAVA code. The first character is incremented by one (+1) and the last character is decremented by one (-1). This changes the identity of the code which is unknown to everyone else.

Step 3:

Then the converted code is padded using random digits generated through our algorithm using a simple code like RAND in JAVA. This padding puts a random digit in front and in the rear of the code which makes it a 4 digit code. These randomly generated digits are saved and only used during the decoding of the character. This changes the length of the converted code and makes it much more secured.

Step 4:

This is the final step to increase the security of the chat system and make the character string highly secured. Here the converted code is taken again and divided using a unique prime number which is 999999989. This gives a different quotient result very time which gives it a very unique identity.

**IMPLEMENTATION**

SERVER SIDE CODE:

import java.net.\*;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import java.util.Scanner;

import java.io.\*;

public class GreetingServer extends Thread

{

String test;

Scanner s= new Scanner(System.in);

private ServerSocket serverSocket;

public GreetingServer(int port) throws IOException

{

serverSocket = new ServerSocket(port);

serverSocket.setSoTimeout(10000);

}

public void run()

{

while(true)

{

try

{

System.out.println("Waiting for client on port " +

serverSocket.getLocalPort() + "...");

Socket server = serverSocket.accept();

System.out.println("Just connected to "

+ server.getRemoteSocketAddress()+"\n\n\n");

do

{

DataInputStream in =

new DataInputStream(server.getInputStream());

System.out.print("Client : " );

test=in.readUTF();

test=decrypt(test);

System.out.println(test);

DataOutputStream out =

new DataOutputStream(server.getOutputStream());

System.out.print("\nServer : ");

test=s.nextLine();

if(test.equalsIgnoreCase("exit"))

{

server.close();

System.out.print("\n\nGoodBye\n\n");

return;

}

test=encrypt(test);

out.writeUTF(test);

}while(true);

}catch(SocketTimeoutException s)

{

System.out.println("Socket timed out!");

break;

}catch(IOException e)

{

System.out.print("\n\n\nClient Disconnected, GoodBye\n\n");

//e.printStackTrace();

break;

}

}

}

public static void main(String [] args)

{

int port = Integer.parseInt(args[0]);

try

{

Thread t = new GreetingServer(port);

t.start();

}catch(IOException e)

{

e.printStackTrace();

}

}

public static String encrypt(String str) {

//Scanner s = new Scanner(System.in);

Random r = new Random();

//System.out.print("Enter Data to be Sent : ");

//String str= s.nextLine();

double x;

str=str.toUpperCase();

char[] charArray = str.toCharArray();

List<Double> intArray = new ArrayList<Double>();

for(int i=0;i<str.length();i++)

{

x=(int)charArray[i];

x-=23;

x+=9;

x\*=10;

x+=(r.nextInt(9)\*1000)+r.nextInt(9);

x=99999989/x;

intArray.add(x);

}

//System.out.println("\n------------------ENCRYPTED KEY-----------------\n\n\n\n");

String result = new String();

int i=0;

while(i<intArray.size())

{

result=result+intArray.get(i).toString();

result+="#";

i++;

}

//System.out.println(result+"\n\nSending Data........\n\n\n");

return result;

}

static public String decrypt(String str) {

//String str ;

char x ;

String result = new String();

String letter = new String();

//Scanner s = new Scanner(System.in);

// str=s.nextLine();

char[] charArray = str.toCharArray();

int i=-1;

double num = 0;

i=0;

while(i<charArray.length)

{

letter="";

for(;i<charArray.length;i++)

{

if(charArray[i]=='#')

break;

letter+=charArray[i];

}

i++;

try {

num=Double.parseDouble(letter);

}catch (NumberFormatException e){

System.out.println("not a number");

}

num=99999989/num;

num=(int)num%1000;

num=num/10;

num=num-9;

num+=(int)23;

x=(char)num;

result=result+x;

}

return result;

}

}

CLIENT SIDE CODE:

**import** java.net.\*;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Random;

**import** java.util.Scanner;

**import** java.io.\*;

**public** **class** GreetingClient

{

**public** **static** **void** main(String [] args)

{

Scanner s = **new** Scanner(System.*in*);

String serverName = args[0];

String p=**null**;

**int** port = Integer.*parseInt*(args[1]);

**try**

{

System.*out*.println("Connecting to " + serverName

+ " on port " + port);

Socket client = **new** Socket(serverName, port);

System.*out*.println("Just connected to "

+ client.getRemoteSocketAddress()+"\n\n\n");

**do**

{

OutputStream outToServer = client.getOutputStream();

DataOutputStream out =

**new** DataOutputStream(outToServer);

System.*out*.print("Client : ");

p=s.nextLine();

**if**(p.equalsIgnoreCase("exit"))

{

client.close();

**break**;

}

p=*encrypt*(p);

out.writeUTF(p);

InputStream inFromServer = client.getInputStream();

DataInputStream in =

**new** DataInputStream(inFromServer);

String test = in.readUTF();

test=*decrypt*(test);

System.*out*.println("Server : " + test);

}**while**(**true**);

}**catch**(IOException e)

{

System.*out*.println("\n\n server says goodbye!! ");

//e.printStackTrace();

}

}

**static** **public** String decrypt(String str) {

//String str ;

**char** x ;

String result = **new** String();

String letter = **new** String();

//Scanner s = new Scanner(System.in);

// str=s.nextLine();

**char**[] charArray = str.toCharArray();

**int** i=-1;

**double** num = 0;

i=0;

**while**(i<charArray.length)

{

letter="";

**for**(;i<charArray.length;i++)

{

**if**(charArray[i]=='#')

**break**;

letter+=charArray[i];

}

i++;

**try** {

num=Double.*parseDouble*(letter);

}**catch** (NumberFormatException e){

System.*out*.println("not a number");

}

num=99999989/num;

num=(**int**)num%1000;

num=num/10;

num=num-9;

num+=(**int**)23;

x=(**char**)num;

result=result+x;

}

**return** result;

}

**public** **static** String encrypt(String str) {

//Scanner s = new Scanner(System.in);

Random r = **new** Random();

//System.out.print("Enter Data to be Sent : ");

//String str= s.nextLine();

**double** x;

str=str.toUpperCase();

**char**[] charArray = str.toCharArray();

List<Double> intArray = **new** ArrayList<Double>();

**for**(**int** i=0;i<str.length();i++)

{

x=(**int**)charArray[i];

x-=23;

x+=9;

x\*=10;

x+=(r.nextInt(9)\*1000)+r.nextInt(9);

x=99999989/x;

intArray.add(x);

}

//System.out.println("\n------------------ENCRYPTED KEY-----------------\n\n\n\n");

String result = **new** String();

**int** i=0;

**while**(i<intArray.size())

{

result=result+intArray.get(i).toString();

result+="#";

i++;

}

//System.out.println(result+"\n\nSending Data........\n\n\n");

**return** result;

}

}

OUTPUT:

**<SERVER SIDE>**

Waiting for client on port 6606...

Just connected to /169.254.227.216:29714

Client : HELLO SERVER

Server : hello client

Client : HELLO AGAIN

Server : this is nice

Client : YES, THIS IS GREAT

Server : exit

GoodBye

**<Client Side>**

Connecting to 169.254.229.103 on port 6606

Just connected to /169.254.229.103:6606

Client : hello server

Server : HELLO CLIENT

Client : hello again

Server : THIS IS NICE

GoodBye

**ENCRYPTION-DECRYPTION**

CODE:

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Random;

**import** java.util.Scanner;

**public** **class** Finalclass {

**static** **public** **void** decrypt(String str) {

System.*out*.println("\n\nRecieved Data....\n");

System.*out*.print(str+"\n\n\n\nDecrypting now......\n\n\n\nDecoded String : ");

//String str ;

**char** x ;

String result = **new** String();

String letter = **new** String();

//Scanner s = new Scanner(System.in);

// str=s.nextLine();

**char**[] charArray = str.toCharArray();

**int** i=-1;

**double** num = 0;

i=0;

**while**(i<charArray.length)

{

letter="";

**for**(;i<charArray.length;i++)

{

**if**(charArray[i]=='#')

**break**;

letter+=charArray[i];

}

i++;

**try** {

num=Double.*parseDouble*(letter);

}**catch** (NumberFormatException e){

System.*out*.println("not a number");

}

num=99999989/num;

num=(**int**)num%1000;

num=num/10;

num=num-9;

num+=(**int**)23;

x=(**char**)num;

result=result+x;

}

System.*out*.println(result);

}

**public** **static** **void** main(String[] args) {

Scanner s = **new** Scanner(System.*in*);

Random r = **new** Random();

System.*out*.print("Enter Data to be Sent : ");

String str= s.nextLine();

**double** x;

str=str.toUpperCase();

**char**[] charArray = str.toCharArray();

List<Double> intArray = **new** ArrayList<Double>();

**for**(**int** i=0;i<str.length();i++)

{

x=(**int**)charArray[i];

x-=23;

x+=9;

x\*=10;

x+=(r.nextInt(9)\*1000)+r.nextInt(9);

x=99999989/x;

intArray.add(x);

}

System.*out*.println("\n------------------ENCRYPTED KEY-----------------\n\n\n\n");

String result = **new** String();

**int** i=0;

**while**(i<intArray.size())

{

result=result+intArray.get(i).toString();

result+="#";

i++;

}

System.*out*.println(result+"\n\nSending Data........\n\n\n");

*decrypt*(result);

}

}

OUTPUT:

Enter Data to be Sent : HELLO

------------------ENCRYPTED KEY-----------------

13190.87046563778#28105.67425519955#38124.2809759817#61652.274352651046#60459.48548972188#

Sending Data........

Recieved Data....

13190.87046563778#28105.67425519955#38124.2809759817#61652.274352651046#60459.48548972188#

Decrypting now......

Decoded String: HELLO

Screen shot:

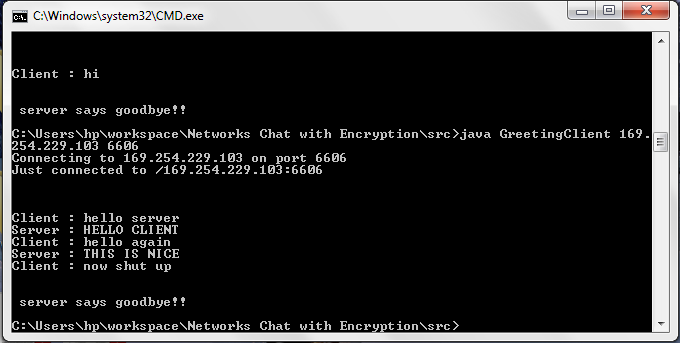


Figure 10: Client side output

**Testing and Analysis**

The analysis of the code algorithm to execute a secured chat system to run properly and transferring data between the server and the client without being interpreted by any other third party is our prime target. The code is built to secure a message sent from the sender to the receiver by interpreting and converting it into a unique identity which is only understood by the receiver client. To run this highly secured process perfectly we run a set of tests and analyse the working of the chat system.

An IP address is provided to the client and the server to connect together using sockets providing a Network Interface between them which helps them to transfer data to and fro. Socket and can be regarded as two program in one of the communication link endpoint, a program will write a message in the Socket, the Socket in this period will send the information to another Socket, which makes this period of information to be transmitted to another applications, this communication process is shown below:



Figure 11

There are four basic server classes: TCP Server uses the Internet TCP protocol, which provides for continuous streams of data between the client and server. UDP Server uses datagrams, which are discrete packets of information that may arrive out of order or be lost while in transit.

These four classes process requests *synchronously*; each request must be completed before the next request can be started. This isn’t suitable if each request takes a long time to complete, because it requires a lot of computation, or because it returns a lot of data which the client is slow to process. The solution is to create a separate process or thread to handle each request; the ForkingMixIn and ThreadingMixIn mix-in classes can be used to support asynchronous behaviour.

Creating a server requires several steps. First, you must create a request handler class by sub-classing the BaseRequestHandler class and overriding its handle() method; this method will process incoming requests. Second, you must instantiate one of the server classes, passing it the server’s address and the request handler class .Finally, call handle\_request() or serve\_forever() method of the server object to process one or many requests.

When inheriting from ThreadingMixIn for threaded connection behavior, you should explicitly declare how you want your threads to behave on an abrupt shutdown. The ThreadingMixIn class defines an attribute *daemon\_threads*, which indicates whether or not the server should wait for thread termination.

You should set the flag explicitly if you would like threads to behave autonomously; the default is False, meaning that Python will not exit until all threads created by ThreadingMixIn have exited.

Server classes have the same external methods and attributes, no matter what network protocol they use.

There are five classes in an inheritance diagram, four of which represent synchronous servers of four types:

+------------+

| BaseServer |

+------------+

|

v

+-----------+ +------------------+

| TCPServer |------->| UnixStreamServer |

+-----------+ +------------------+

|

v

+-----------+ +--------------------+

| UDPServer |------->| UnixDatagramServer |

+-----------+ +--------------------+

Figure 12

Firstly, server sockets are generally bound to well known names (ports, in this case) and they establish themselves with listen(). That is where the real difference happens, as client sockets establish themselves with connect(). Calling listen() on a socket causes the kernel's TCP/IP implementation to begin accepting connections sent to the socket's bound name (port). This will happen whether or not you ever call accept().

accept() simply gives your server a way to access and interact with the client sockets that have connected to your listening socket.

At a low level sockets are just sockets regardless of whether they are being used in a server or client application. The difference between the two lies in the system calls each kind of application makes.

Server sockets will call bind() to be associated with a port. They want to be associated with a port so that other programs know where to reach them. Client sockets *can* call bind() but almost never do because there is not much point. If a socket doesn't call bind() the OS will just choose an ephemeral port for it, which is fine for clients because they are doing the calling; no one *needs* to call them.

Server sockets call listen(). This was explained pretty well in the other answers.

Server sockets call accept() and I think this is the crux of your question because it is a bit mysterious at first. The important thing to grasp is that in calling accept() the kernel will pass back a *new* socket. It is now separate from the original listening socket and is what your server will use to communicate with its peer(s).

The key in understanding how the listening socket continues to listen while the accepted connection is doing its thing is in understanding that tcp connections depend on a 4-tuple of (1) local address (2) local port (3) foreign address (4) foreign port. These define a unique connection. Before accept() passed back the new socket the kernel used these values to create various structures so that in collaboration with the tcp/ip stack all traffic with this tuple will go to the connected socket. Even though your server may have a thousand connections with local address 192.168.1.100 port 80, the client combination of address and port will always be different and thus the tuple is always unique.

**Conclusions and Future Enhancements**

The most vital blessing to the human race is the ability to communicate impeccably with ease of variation of languages and many other modes of transferring data from one to the other.

Since the evolution of mankind, came along the evolution of communication. As it started with the Hunters talking through pictography and sign languages to the invention of cloud computing.

With increasing methods people find to communicate comes the threat of hacks and cyber crimes. But, as the great saying goes…”Prevention is better than cure”.

With the immense increase in the rate of invention of the new upcoming techniques for the security of the encryption and decryption

After our research and implementation of the project regarding cloud server creation and connectivity of the client to establish a secure connection for the purpose of confidential communication through concepts of symmetric encryption technique that is custom made and can be used for a decent level of security.

The connectivity of socket programming relies on the Internet protocol (IP) addressing of the server and the client with the additional attachment of the ports to establish the link of both the applications. Since the port address is exclusive for each connectable application, no third party can join the conversation unless there is a man in the middle who may be fiddling with the data as an active attacker or maybe using the data as it is to be manipulated later and not fondling with the ongoing data communication like a passive attacker.

We, by our very own devised encryption-decryption symmetric algorithm, aim to make the data useless for the man in the middle as it is encoded or encrypted as integers, which might not give out any direct clues to the reader.

The technique used is a Four-Level encryption technique including randomization, division by a large prime number, padding and also arithmetic manipulation to get the desired encrypted code.

Decryption of the encrypted data is also a long process including complementing of each and every method used for randomization, division by a large prime number, padding and the arithmetic manipulation.

All of this happens in the reverse process as we are complementing the process of encryption, i.e., we are decrypting or decoding the message into a receiver understandable form so that the receiver can understand the message he’ll be getting through the sender who initially encrypted or encoded the message to make it secure through the communication lines in case of an external attack be it active or passive.

**References:**

* ***“The Design Method of Network Chat System Based on Socket and Cloud Computing”, IEEE paper published in IEEE explore in year 2012***
* [**http://www.ijcaonline.org/journal/number15/pxc387502.pdf**](http://www.ijcaonline.org/journal/number15/pxc387502.pdf)
* [**http://www.ijarcce.com/upload/2013/august/29-o-sumit%20chaudhary%20-Encryption%20and%20Decryption%20Algorithm%20using%20ASCII%20values%20with.pdf**](http://www.ijarcce.com/upload/2013/august/29-o-sumit%20chaudhary%20-Encryption%20and%20Decryption%20Algorithm%20using%20ASCII%20values%20with.pdf)
* [**http://www.buyya.com/java/Chapter13.pdf**](http://www.buyya.com/java/Chapter13.pdf)
* [**http://stackoverflow.com**](http://stackoverflow.com)