Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 4, April 2015, pg.761 – 767

RESEARCH ARTICLE

Security Enhancement in Distributed Networking

Praveen Balda, Sh. Matish Garg

Distributed Networking is a distributed computing network system, said to be "distributed" when the computer programming and the data to be worked on are spread out over more than one computer.
Usually, this is implemented over a network.
Prior to the emergence of low-cost desktop computer power, computing was generally centralized to one computer.
Although such centers still exist, distribution networking applications and data operate more efficiently over a mix of desktop workstations, local area network servers, regional servers, Web servers, and other servers.
One popular trend is client/server computing. This is the principle that a client computer can provide certain capabilities for a user and request others from other computers that provide services for the clients. (The Web's Hypertext Transfer Protocol is an example of this idea.)
Enterprises that have grown in scale over the years and those that are continuing to grow are finding it extremely challenging to manage their distributed network in the traditional client/server computing model.
The recent developments in the field of cloud computing has opened up new possibilities. Cloud-based networking vendors have started to sprout offering solutions for enterprise distributed networking needs. Whether it turns out to revolutionize the distributed networking space or turns out to be another craze remains to be seen.

Properties of Distributed System

The word *distributed* in terms such as "distributed system", "distributed programming", and "distributed algorithm" originally referred to computer networks where individual computers were physically distributed within some geographical area. The terms are nowadays used in a much wider sense, even referring to autonomous processes that run on the same physical computer and interact with each other by message passing. While there is no single definition of a distributed system, the following defining properties are commonly used:

- There are several autonomous computational entities, each of which has its own local memory.
- The entities communicate with each other by message passing.

The computational entities are called *computers* or *nodes*.

A distributed system may have a common goal, such as solving a large computational problem. Alternatively, each computer may have its own user with individual needs, and the purpose of the

distributed system is to coordinate the use of shared resources or provide communication services to the users.

Other typical properties of distributed systems include the following:

- The system has to tolerate failures in individual computers.
- The structure of the system (network topology, network latency, number of computers) is not known in advance, the system may consist of different kinds of computers and network links, and the system may change during the execution of a distributed program.
- Each computer has only a limited, incomplete view of the system. Each computer may know only one part of the input.

Parallel and distributed computing

Distributed systems are groups of networked computers, which have the same goal for their work. The terms "concurrent computing", "parallel computing", and "distributed computing" have a lot of overlap, and no clear distinction exists between them.

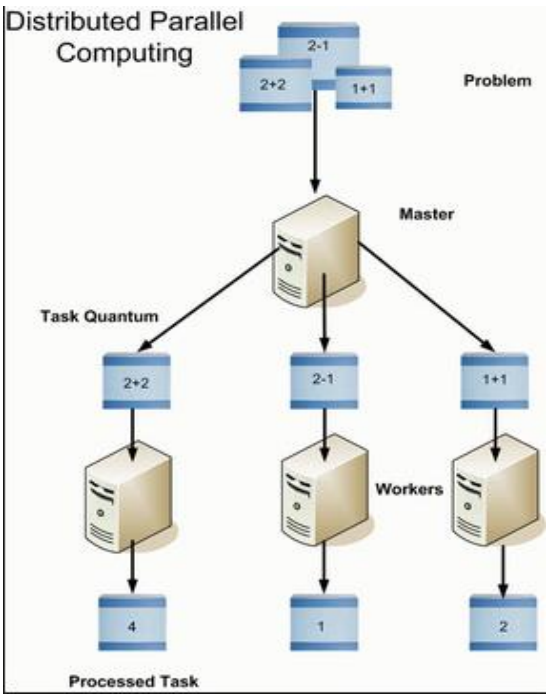

Fig. 1

The same system may be characterized both as "parallel" and "distributed"; the processors in a typical distributed system run concurrently in parallel. Parallel computing may be seen as a particular tightly coupled form of distributed computing, and distributed computing may be seen as a loosely coupled form of parallel computing. Nevertheless, it is possible to roughly classify concurrent systems as "parallel" or "distributed" using the following criteria:

- In parallel computing, all processors may have access to a shared memory to exchange information between processors.
- In distributed computing, each processor has its own private memory (distributed memory). Information is exchanged by passing messages between the processors.

Cryptography prior to the modern age was effectively synonymous with encryption, the conversion of information from a readable state to apparent nonsense. The originator of an encrypted message shared the decoding technique needed to recover the original information only with intended recipients, thereby precluding unwanted persons to do the same. Since World War I and the advent of the computer, the methods used to carry out cryptology have become increasingly complex and its application more widespread.

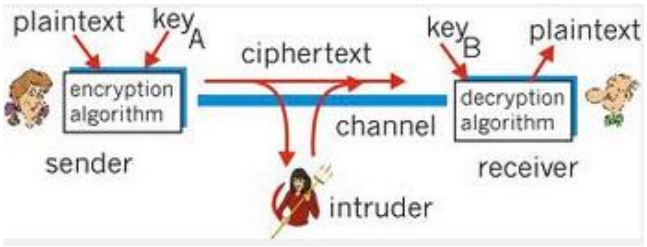

Fig. 2

Cryptography

Modern cryptography is heavily based on mathematical theory and computer science practice; cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary. It is theoretically possible to break such a system but it is infeasible to do so by any known practical means. These schemes are therefore termed computationally secure; theoretical advances (e.g., improvements in integer factorization algorithms) and faster computing technology require these solutions to be continually adapted. There exist information-theoretically secure schemes that provably cannot be broken even with unlimited computing power—an example is the one-time pad—but these schemes are more difficult to implement than the best theoretically breakable but computationally secure mechanisms.

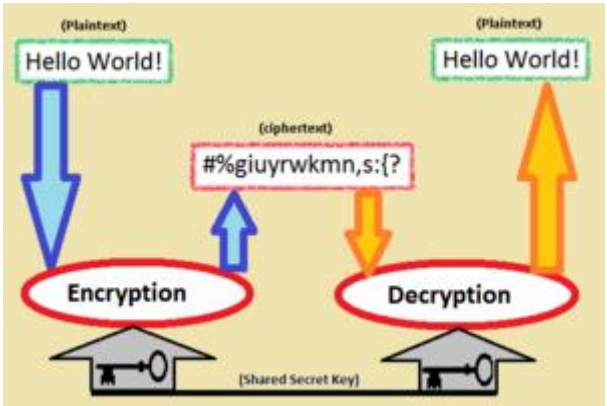

Fig 3. Encryption and Decryption of text

The goals of cryptography are:

1. *Access Control* is the ability to limit and control the access to the system and application.
2. *Confidentiality* requires that a cryptanalyst will not be able to determine plain text from intercepted cipher text. There are numerous approaches to providing confidentiality, ranging from physical protection to mathematical algorithms which provide data unintelligible.
3. *Data integrity* is a service which addresses the unauthorized alteration of data. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties. Data manipulation includes such things as insertion, deletion, and substitution.
4. *Authentication* is to assure the recipient that the message is from the source that it claims to be from [6]. Two parties entering into a communication should identify each other. Information delivered over a channel should be authenticated. Cryptography is usually subdivided into two major classes: *entity authentication* and *data origin authentication*.
5. *Data Integrity* addresses the unauthorized modification of data. One must have the ability to detect data manipulation by unauthorized parties such as insertion, deletion and substitution to ensure data integrity.
6. *Non-repudiation* is a service which prevents an entity from denying previous commitments or actions. When disputes arise due to an entity denying that certain actions were taken, a means to resolve the situation is necessary.
7. *Availability* requires that computer system possessions be available to authorized parties when needed.

As well as being aware of cryptographic history, cryptographic algorithm and system designers must also sensibly consider probable future developments while working on their designs. For instance, continuous improvements in computer processing power have increased the scope of brute-force attacks, thus when specifying key lengths, the required key lengths are similarly advancing. The potential effects of quantum computing are already being considered by some cryptographic system designers; the announced imminence of small implementations of these machines may be making the need for this preemptive caution rather more than merely speculative.

Basic algorithm and terminology

RSA encryption and decryption are essentially mathematical operations. They are what are termed *exponentiation*, *modulo* a particular number. Because of this, RSA keys actually consist of numbers involved in this calculation, as follows:

- The public key consists of the modulus and a public exponent;
- The private key consists of that same modulus plus a private exponent.

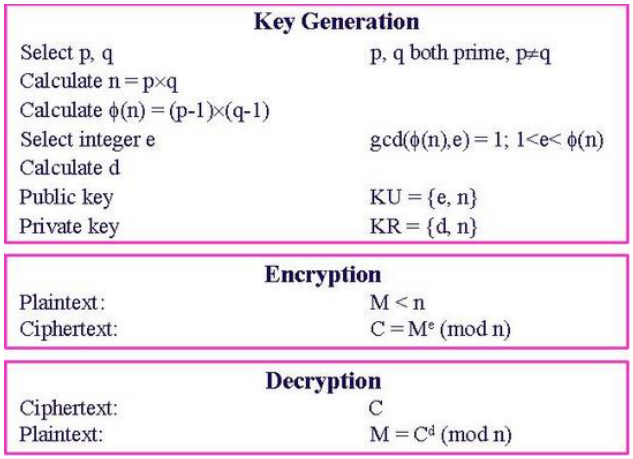

Fig. 4

Encrypting a String with DES

This example implements a class for encrypting and decrypting strings using DES. The class is created with a key and can be used repeatedly to encrypt and decrypt strings using that key.

```
public class DesEncrypter {
   Cipher ecipher;
   Cipher dcipher;

   DesEncrypter(SecretKey key) {
      try {
         ecipher = Cipher.getInstance("DES");
         dcipher = Cipher.getInstance("DES");
         ecipher.init(Cipher.ENCRYPT_MODE, key);
         dcipher.init(Cipher.DECRYPT_MODE, key);

      } catch (javax.crypto.NoSuchPaddingException e) {
      } catch (java.security.NoSuchAlgorithmException e) {
      } catch (java.security.InvalidKeyException e) {
      }
   }

   public String encrypt(String str) {
      try {
        // Encode the string into bytes using utf-8
         byte[] utf8 = str.getBytes("UTF8");

         // Encrypt
         byte[] enc = ecipher.doFinal(utf8);

   // Encode bytes to base64 to get a string
       return new sun.misc.BASE64Encoder().encode(enc);
      } catch (javax.crypto.BadPaddingException e) {
      } catch (IllegalBlockSizeException e) {
      } catch (UnsupportedEncodingException e) {
      } catch (java.io.IOException e) {
```



```
        }
        return null;
    }

    public String decrypt(String str) {
        try {
            // Decode base64 to get bytes
            byte[] dec = new sun.misc.BASE64Decoder().decodeBuffer(str);

            // Decrypt
            byte[] utf8 = dcipher.doFinal(dec);

            // Decode using utf-8
            return new String(utf8, "UTF8");
        } catch (javax.crypto.BadPaddingException e) {
        } catch (IllegalBlockSizeException e) {
        } catch (UnsupportedEncodingException e) {
        } catch (java.io.IOException e) {
        }
        return null;
    }
}
```

Here's an example that uses the class:

```
try {
    // Generate a temporary key. In practice, you would save this key.
    // See also Encrypting with DES Using a Pass Phrase.
    SecretKey key = KeyGenerator.getInstance("DES").generateKey();

    // Create encrypter/decrypter class
    DesEncrypter encrypter = new DesEncrypter(key);

    // Encrypt
    String encrypted = encrypter.encrypt("Don't tell anybody!");

    // Decrypt
String decrypted = encrypter.decrypt(encrypted);
} catch (Exception e) {
}
```

OBJECTIVE

The aim is to create customized java code for Encryption and decryption that should be more secure and efficient.

- Some time information to be send are multiple and merged using delimiter into plain text then at the time of decryption plain text is split again in multiple pieces of information.

- Suppose there are different users with different privilege level and selected pieces of information are to be delivered to particular user than there is need to alter decryption Module so that only authorized user could get authorized information.

- To protect information from cryptanalyst IP Filter would be attached in decryption module

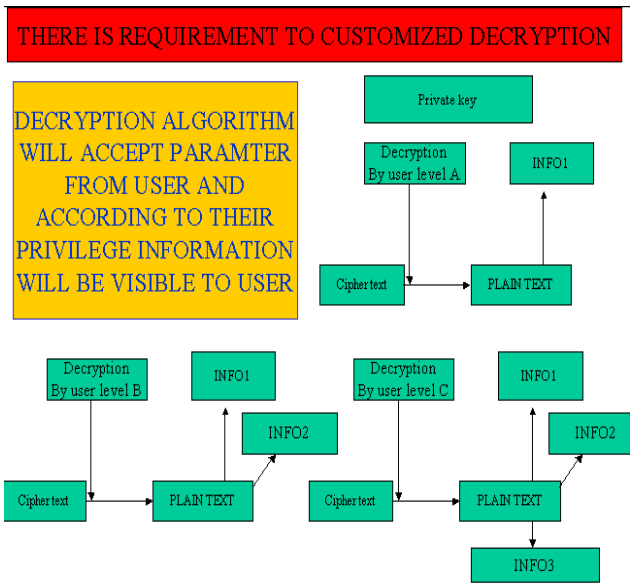

Fig 5.

References

- Network Security (William Stalling)
- Computer Network(TanenBaum)
- Data Communication (Forouzan)
- Networking Complete Reference(TMH)
- Java Network programming(ORELLAY)
- JAVA BLACK BOOK(DREAM TECH)
- JAVA COMPLETE REFERENCE(TMH)
- FIREWALL 24/7(BPB)

