Workload Characterization of Elliptic Curve Cryptography and Other Network Security Algorithms for Constrained Environments

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

in

Information Technology

under the Supervision of

Dr. Hemraj Saini

By

Mitesh Gupta - 111434

to

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

CERTIFICATE

This is to certify that project report entitled “Workload Characterization of Elliptic Curve Cryptography and other Network Security algorithms for Constrained Environments”, submitted by Mitesh Gupta (111434) in partial fulfillment for the award of degree of Bachelor of Technology in Information Technology to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date: **Dr. Hemraj Saini**

Assistant Professor (Senior Grade)

ACKNOWLEDGEMENT

It gives me a great sense of pleasure to present the project report entitled **“Workload Characterization of Elliptic Curve Cryptography and Other Network Security Algorithms for Constrained Environments”**. It is my privilege to express my sincerest regards to my project supervisor, **Dr. Hemraj Saini**, for his valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of the project.

I deeply express my sincere thanks to the **Prof. R. M. K. Sinha**, Dean and Head of Department, Department of Information Technology and **Brig. (Retd.) S. P. Ghrera**, Head of Department, Department of Computer Science and Engineering, for encouraging and allowing me to present this project.

I take this opportunity to thank all the lecturers who have directly or indirectly helped in this project. I pay my respect and love to my parents and all other family members for their love and encouragement throughout my career. Last but not the least, I express my thanks to my friends for their cooperation and support.

Date: **Mitesh Gupta**

TABLE OF CONTENTS

LIST OF FIGURES

List of Tables

ABSTRACT

Today, security is one of the foremost concerns in the transmission of data over the internet or over any other network. Also, in recent years, there are great speculations regarding cryptographic algorithms which are suitable for constrained environments, like mobile devices, where computing resources, power availability, communication bandwidth and available memory are limited, as the number of mobile users are increasing at a very fast pace due to which m-commerce is on boom and today accounts for around 50% of all the e-commerce traffic and this percentage is expected to increase in future.

In this project, I will select a public key cryptosystem suitable for such environment and study their workload characteristics. Particularly, I will study and discuss about elliptic curve cryptography algorithms, which is an emerging public key cryptographic system for constrained environments. This allows fast software implementations because of reduced key sizes and computational efficiency, while preserving the same security levels as were offered by previous integer-based public-key algorithms. I will characterize the operations needed by elliptic curve cryptography for different key sizes and different levels of software optimization. I will try to show that this algorithm can be implemented efficiently using very less resources.

1. INTRODUCTION TO NETWORK SECURITY AND CRYPTOGRAPHY

In this age of universal electronic connectivity, of viruses and hackers, of electronic eavesdropping and electronic fraud, there is indeed no time at which security does not matter. Two trends have come together to make the topic “Cryptography” of vital interest. First, the explosive growth in computer systems and their interconnections via networks has increased the dependence of both organizations and individuals on the information stored and communicated using these systems. This, in turn, has led to a heightened awareness of the need to protect data and resources from disclosure, to guarantee the authenticity of data and messages, and to protect systems from network-based attacks. Second, the disciplines of cryptography and network security have matured, leading to the development of practical, readily available applications to enforce network security [10].

In this project, I will be mainly focusing on the public key cryptosystems, which are mainly used for key distribution and digital signatures. But before that, I would like to give a brief theoretical background about cryptography.

1.1 Some Basic Definitions

Computer Security: The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunication). Basically, it is protection of information in potentially hostile environments.

Cryptography: It is the key technology for achieving information security in communications, computer systems, electronic commerce, and in the emerging information society. It is a science that applies mathematics and logic to design complex methods to encrypt data.

Threat: A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit a vulnerability.

Attack: An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system [14].

1.2 Network Security

Network security consists of the provisions and policies adopted by a network administrator to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources. Network security involves the authorization of access to data in a network, which is controlled by the network administrator. Users choose or are assigned an ID and password or other authenticating information that allows them access to information and programs within their authority. Network security covers a variety of computer networks, both public and private, that are used in everyday jobs conducting transactions and communications among businesses, government agencies and individuals. Networks can be private, such as within a company, and others which might be open to public access. Network security is involved in organizations, enterprises, and other types of institutions. It does as its title explains: It secures the network, as well as protecting and overseeing operations being done.

ITU-T (The International Telecommunication Union – Telecommunication Standardisation Sector) Recommendation X.800, Security Architecture for OSI, defines such a systematic approach.4 The OSI security architecture is useful to managers as a way of organizing the task of providing security. Furthermore, because this architecture was developed as an international standard, computer and communications vendors have developed security features for their products and services that relate to this structured definition of services and mechanisms [14].

The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly as

- **Security Attack**
  Any action that compromises the security of information owned by an organization.
- **Security Mechanism**

A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack.

- **Security Service**

  A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service.

1.2.1 Security Attacks

Security Attacks can be classified in terms of passive attacks and active attacks [14].

1. **Passive attacks**

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

   - *Release of message attack*

     This type of attack is easily understood. A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.

   - *Traffic analysis*

     Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place.

   Passive attacks are very difficult to detect, because they do not involve any alteration of the data. Typically, the message traffic is sent and received in an apparently normal fashion, and neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern. However, it is feasible to prevent the success of these attacks, usually by means of encryption. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

2. **Active attacks**

   Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories:

   - *Masquerade*

     This takes place when one entity pretends to be a different entity. A masquerade attack usually includes one of the other forms of active attack.
   - *Replay*

     This involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.
   - *Modification of messages*

     This simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect.
   - *Denial-of-Service*

     The denial of service prevents or inhibits the normal use or management of communications facilities. This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service). Another form of service denial is the disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance.

   Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are available to prevent their success.

   On the other hand, it is quite difficult to prevent active attacks absolutely because of the wide variety of potential physical, software, and network vulnerabilities. Instead, the goal is to detect active attacks and to recover from any disruption or delays caused by them. If the detection has a deterrent effect, it may also contribute to prevention.

1.2.2 Security Services

X.800 defines a security service as a service that is provided by a protocol layer of communicating open systems and that ensures adequate security of the systems or of data transfers [14].

X.800 divides security services into five categories:

1. **Authentication**

   The assurance that the communicating entity is the one that it claims to be.

   - *Peer Entity Authentication*

     Used in association with a logical connection to provide confidence in the identity of the entities connected.

   - *Data Origin Authentication*

     In a connectionless transfer, provides assurance that the source of received data is as claimed.

2. **Access Control**

   The prevention of unauthorized use of a resource (i.e., this service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).

3. **Data confidentiality**

   The protection of data from unauthorized disclosure.

   - *Connection Confidentiality*

     The protection of all user data on a connection.

   - *Connectionless Confidentiality*

     The protection of all user data in a single data block

   - *Selective-Field Confidentiality*

     The protection of all user data in a single data block

   - *Traffic-Flow Confidentiality*

     The protection of the information that might be derived from observation of traffic flows.

4. **Data Integrity**

   The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).

   - *Connection Integrity with Recovery*

Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.

- *Connection Integrity without Recovery*
  As above, but provides only detection without recovery.
- *Selective-Field Connection Integrity*
  Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.
- *Connectionless Integrity*
  Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.
- *Selective-Field Connectionless Integrity*
  Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.

5. **Non-Repudiation**
   Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.
   - *Nonrepudiation, Origin*
     Proof that the message was sent by the specified party.
   - *Nonrepudiation, Destination*
     Proof that the message was received by the specified party.

1.2.3 Security Mechanisms

The mechanisms are divided into those that are implemented in a specific protocol layer, such as TCP or an application-layer protocol, and those that are not specific to any particular protocol layer or security service.

However, X.800 distinguishes between reversible encipherment mechanisms and irreversible encipherment mechanisms. A *reversible encipherment mechanism* is simply an encryption algorithm that allows data to be encrypted and subsequently decrypted. *Irreversible encipherment mechanisms* include hash algorithms and message authentication codes, which are used in digital signature and message authentication applications [14].

1. **Specific Security Mechanisms**

   May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.

   - *Encipherment*

     The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.
   - *Digital Signature*

     Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).
   - *Access Control*

     A variety of mechanisms that enforce access rights to resources.
   - *Data Integrity*

     A variety of mechanisms used to assure the integrity of a data unit or stream of data units.
   - *Authentication Exchange*

     A mechanism intended to ensure the identity of an entity by means of information exchange.
   - *Traffic Padding*

     The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.
   - *Routing Protocol*

     Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.
   - *Notarization*

     The use of a trusted third party to assure certain properties of a data exchange.

2. **Pervasive Security Mechanisms**

   Mechanisms that are not specific to any particular OSI security service or protocol layer.

   - *Trusted Functionality*

     That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).

   - *Security Label*

     The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.

   - *Event Detection*

     Detection of security-relevant events.

   - *Security Audit Trail*

     Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.

   - *Security Recovery*

     Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

1.2.4 Network Security Model

A model for much of what we will be discussing is captured, in very general terms, in Figure 1. A message is to be transferred from one party to another across some sort of Internet service. The two parties, who are the principals in this transaction, must cooperate for the exchange to take place. A logical information channel is established by defining a route through the Internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals [14].

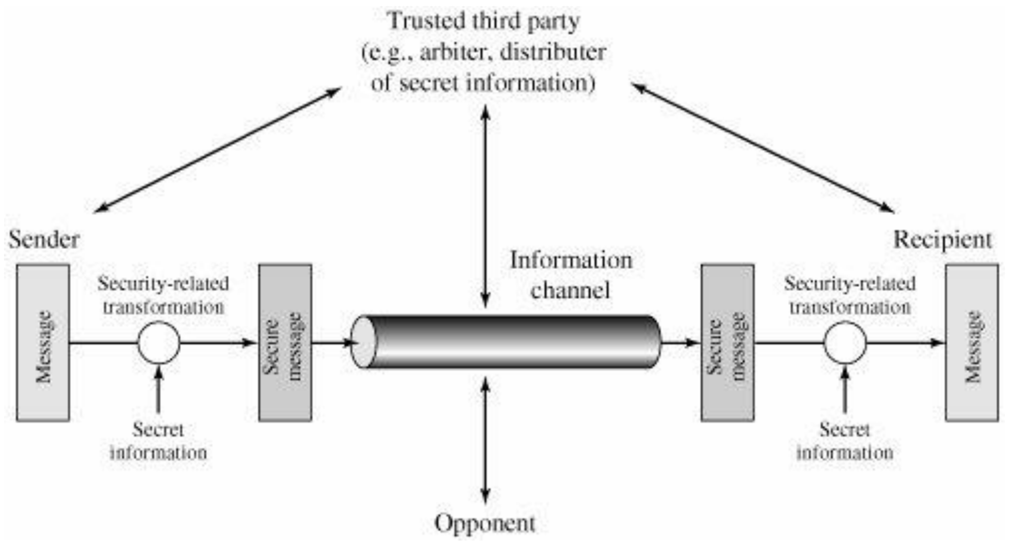


Figure 1 Network Security Model

Security aspects come into play when it is necessary or desirable to protect the information transmission from an opponent who may present a threat to confidentiality, authenticity, and so on. All the techniques for providing security have two components:

- A security-related transformation on the information to be sent. Examples include the encryption of the message, which scrambles the message so that it is unreadable by the opponent, and the addition of a code based on the contents of the message, which can be used to verify the identity of the sender.
- Some secret information shared by the two principals and, it is hoped, unknown to the opponent. An example is an encryption key used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception.

A trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the authenticity of a message transmission [14].

1.3 Cryptography and its Techniques

Cryptography is a method of storing and transmitting data in a particular form so that only those for whom it is intended can read and process it. The term is most often associated with scrambling plaintext into ciphertext (a process called encryption), then back again (known as decryption) [10].

Cryptography can be classified in several ways. In this report, we will categorize cryptography in two main types, based on the number of keys which are used in encryption and decryption process. These types are:

- Symmetric key cryptography
- Public key cryptography

1.3.1 Symmetric Key Cryptography (SKC)

Symmetric encryption, also referred to as conventional encryption or single-key encryption, was the only type of encryption in use prior to the development of public key encryption in the 1970s. It remains by far the most widely used of the two types of encryption.

In this, a single key is used for both encryption and decryption. As shown in Figure 2, the sender uses the key (or some set of rules) to encrypt the plaintext and sends the ciphertext to the receiver. The receiver applies the same key to decrypt the message and recover the plaintext [10].

▶ **Symmetric Cipher Model**

A symmetric encryption scheme has five ingredients [14]:

- *Plaintext*

  This is the original intelligible message or data that is fed into the algorithm as input.
- *Encryption Algorithm*

  The encryption algorithm performs various substitutions and transformations on the plaintext.
- *Secret Key*

  The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.

- *Ciphertext*

  This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.
- *Decryption Algorithm*

  This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

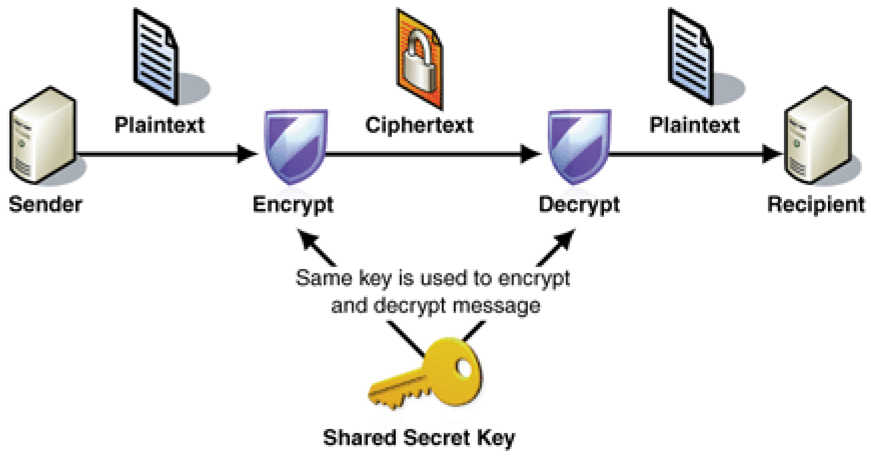


Figure 2 Symmetric Key Cryptography

With this form of cryptography, it is obvious that the key must be known to both the sender and the receiver; that, in fact, is the secret. The biggest difficulty with this approach, of course, is the distribution of the key.

Symmetric 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.

Major Symmetric key cryptographic algorithms in use today are:

- **Data Encryption Standard (DES)**

  The most common SKC scheme used today, DES was 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 is a block-cipher employing a 56-bit key that operates on 64-bit blocks. DES has a complex set of rules and transformations that were designed specifically to yield fast hardware implementations and slow software implementations, although this latter point is becoming less significant today since the speed of computer processors is several orders of magnitude faster today than twenty years ago. IBM also proposed a 112-bit key for DES, which was rejected at the time by the government; the use of 112-bit keys was considered in the 1990s, however, conversion was never seriously considered [10].

  DES is defined in American National Standard X3.92 and three Federal Information Processing Standards (FIPS):

  - FIPS 46-3: DES
  - FIPS 74: Guidelines for Implementing and Using the NBS Data
  - Encryption Standard
  - FIPS 81: DES Modes of Operation

- **Triple-DES (3DES)**

  A variant of DES that employs up to three 56-bit keys and makes three encryption/decryption passes over the block; 3DES is also described in FIPS 46-3 and is the recommended replacement to DES [10].

- **Advanced Encryption Standard (AES)**

  In 1997, NIST initiated a very public, 4-1/2 year process to develop a new secure cryptosystem for U.S. government applications. The result, the Advanced Encryption Standard, became the official successor to DES in December 2001. AES uses an SKC scheme called Rijndael, a block cipher designed by Belgian cryptographers Joan Daemen and Vincent Rijmen. The algorithm can use a variable block length and key length; the latest specification allowed any combination of keys lengths of 128, 192, or 256 bits and blocks of length 128, 192, or 256 bits. NIST initially selected Rijndael in October 2000 and formal adoption as the AES standard came in December 2001. FIPS PUB 197 describes a 128-bit block cipher employing a 128-, 192-, or 256-bit key [10].

1.3.2 Public Key Cryptography (PKC)

Public-key cryptography has been said to be the most significant new development in cryptography in the last 300-400 years. Modern PKC was first described publicly by Stanford University professor Martin Hellman and graduate student Whitfield Diffie in 1976. Their paper described a two-key cryptosystem in which two parties could engage in a secure communication over a non-secure communications channel without having to share a secret key [10].

PKC depends upon the existence of so-called one-way functions, or mathematical functions that are easy to compute whereas their inverse function is relatively difficult to compute. Let me give you two simple examples:

Multiplication vs. factorization: Suppose I tell you that I have two numbers, 9 and 16, and that I want to calculate the product; it should take almost no time to calculate the product, 144. Suppose instead that I tell you that I have a number, 144, and I need you tell me which pair of integers I multiplied together to obtain that number. You will eventually come up with the solution but whereas calculating the product took milliseconds, factoring will take longer because you first need to find the 8 pair of integer factors and then determine which one is the correct pair.

Exponentiation vs. logarithms: Suppose I tell you that I want to take the number 3 to the 6th power; again, it is easy to calculate $3^6 = 729$. But if I tell you that I have the number 729 and want you to tell me the two integers that I used, x and y so that $\log_x 729 = y$, it will take you longer to find all possible solutions and select the pair that I used.

While the examples above are trivial, they do represent two of the functional pairs that are used with PKC; namely, the ease of multiplication and exponentiation versus the relative difficulty of factoring and calculating logarithms, respectively. The mathematical "trick" in PKC is to find a trap door in the one-way function so that the inverse calculation becomes easy given knowledge of some item of information.

Generic PKC employs two keys that are mathematically related although knowledge of one key does not allow someone to easily determine the other key. One key is used to encrypt the plaintext and the other key is used to decrypt the ciphertext. The important point here is that it does not matter which key is applied first, but that both keys are required for the process to work (Figure 3). Because a pair of keys are required, this approach is also called asymmetric cryptography.

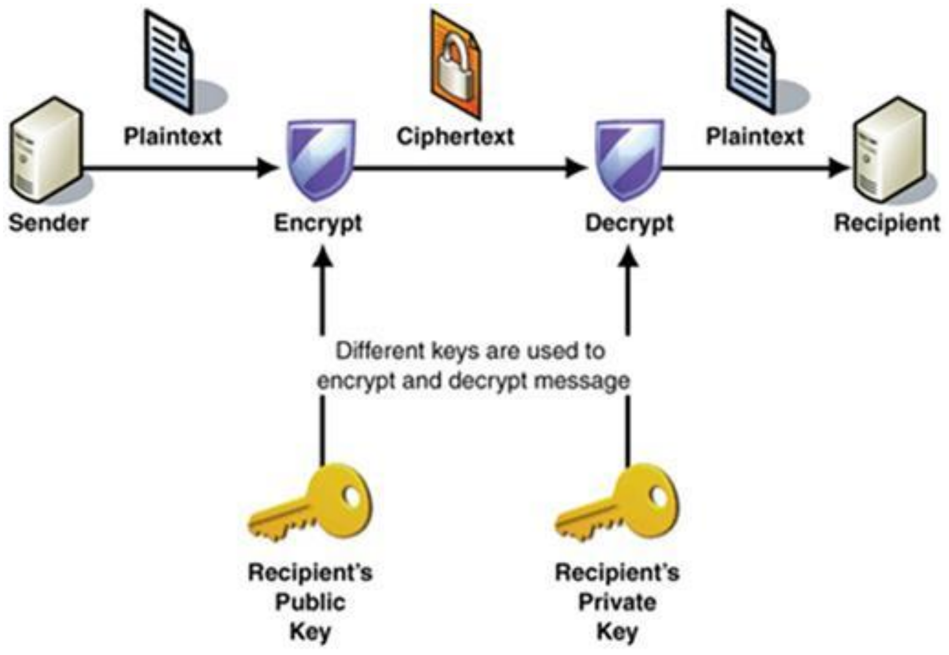


Figure 3 Public Key Cryptography

In PKC, one of the keys is designated the public key and may be advertised as widely as the owner wants. The other key is designated the private key and is never revealed to another party. It is straight forward to send messages under this scheme. Suppose Alice wants to send Bob a message. Alice encrypts some information using Bob's public key; Bob decrypts the ciphertext using his private key. This method could be also used to prove who sent a message; Alice, for example, could encrypt some plaintext with her private key; when Bob decrypts using Alice's public key, he knows that Alice sent the message and Alice cannot deny having sent the message (Nonrepudiation) [10].

▶ Applications of Public Key Cryptography

Before proceeding, we need to clarify one aspect of public-key cryptosystems that is otherwise likely to lead to confusion. Public-key systems are characterized by the use of a cryptographic algorithm with two keys, one held private and one available publicly. Depending on the application, the sender uses either the sender's private key or the receiver's public key, or both, to perform some type of cryptographic function. In broad terms, we can classify the use of public-key cryptosystems into three categories:

- *Encryption/Decryption*
  The sender encrypts a message with the recipient's public key.
- *Digital Signatures*
  The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
- *Key Exchange*
  Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

Some algorithms are suitable for all three applications, whereas others can be used only for one or two of these applications. Table 1 indicates the applications supported by the some of the public-key algorithms.

Table 1 Application for Public-Key Cryptosystems

Algorithm	**Encryption/Decryption**	**Digital Signature**	**Key Exchange**
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes

▶ Requirements for Public Key Cryptography

This cryptosystem depends on a cryptographic algorithm based on two related keys. Diffie and Hellman postulated this system without demonstrating that such algorithms exist. However, they did lay out the conditions that such algorithms must fulfil [14]. Conditions are:

1. It is computationally easy for a party B to generate a pair (public key PU_b, private key PR_b).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext:

$$C = E(PU_b, M)$$

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

$$M = D\ (PR_b, C) = D\ [PR_b, E\ (PU_b, M)]$$

4. It is computationally infeasible for an adversary, knowing the public key, PU_b, to determine the private key, PR_b.
5. It is computationally infeasible for an adversary, knowing the public key, PU_b, and a ciphertext, C, to recover the original message, M.
6. The two keys can be applied in either order:

$$M = D\ [PU_b, E\ (PR_b, M)] = D\ [PR_b, E\ (PU_b, M)]$$

These are formidable requirements, as evidenced by the fact that only a few algorithms (RSA, elliptic curve cryptography, Diffie-Hellman) have received widespread acceptance in the several decades since the concept of public-key cryptography was proposed.

The requirements boil down to the need for a trap-door one-way function. A one-way function3 is one that maps a domain into a range such that every function value has a unique inverse, with the condition that the calculation of the function is easy, whereas the calculation of the inverse is infeasible.

$X = f^{-1}(Y)$	Infeasible
$Y=f(X)$	Easy

2. WORKLOAD CHARACTERISATION

2.1 An Overview

The workload of a system can be defined as the set of all inputs that the system receives from its environment during any given period of time.

Workload parameters or Workload features: Measured quantities, service requests, or resource demands. For example: transaction types, instructions, packet sizes, source-destinations of a packet, and page reference pattern [11].

We can characterize workload at four levels: business, session, function, and protocol. We can view an e-business's workload in a multilayer hierarchical way. Each layer has many features:

- **Business layer:**
  We can examine a business's overall characteristics and how they affect the way users interact with the site in this layer. For an auction site, for example, we would be interested in the number of bids the winner placed, the point at which the winner placed his or her first bid, the closing price's evolution over the auction's life, and the percentage of auctions that have winners.
- **Session layer:**
  We define a session as the sequence of requests a customer makes during a single visit to an e-business site. Thus, the session layer deals with characteristics such as duration (measured in the number of requests per session), navigation patterns within a session, and the buy-to-visit ratio (the probability that a session will result in a sale). Customer behaviour model graphs (CBMGs) and customer visit models (CVMs) are two examples of models used to capture the way customers invoke the various functions an e-business site offers.
- **Function layer:**
  An e-business site offers many functions to customers. An auction site, for example, provides browse, search, register, login, view bid, bid, and sell functions.
- **HTTP request layer:**
  Customers interact with ecommerce sites through the HTTP protocol; characterization at this level deals with the workload's features in terms of HTTP requests.

2.2 Business Layer Characterisation

We analyse the workload at the business level to determine which aspects are common to all sessions and which ones influence the business's profitability and the site's overall revenue throughput (measured here as the number of dollars generated per time unit).

2.3 Session Layer Characterisation

The number of sessions submitted to an e-business site is huge, so grouping, or clustering, similar sessions can help characterize the site's workload. We use two classes of clustering algorithms for this purpose: distance-based and fractal-based. Distance-based algorithms (such as the k-means clustering algorithm) group sessions according to a defined distance between the sessions. If we use a CBMG to represent different sessions, an underlying matrix indicates each session's probabilities of transition between states. We can then use the Euclidean distance between the matrices representing two separate CBMGs for the clustering process. Distance-based clustering has its limitations — for example, all clusters must have regular geometrical shapes. Fractal-based clustering offers an improvement in quality: it forms clusters with any arbitrary shape. Fractal based clustering allocates each point to the cluster whose fractal dimension changes the least by the inclusion of that point. A more thorough discussion of a data set's fractal dimension appears elsewhere. Two important aspects of session layer workload are session length distribution and a multiple time scale analysis of the number of sessions initiated per time unit. The session length distribution's tail (the probability that the length exceeds a given value) for an online bookstore and an auction site. Here, we measure session length as the number of e-business functions invoked per session. Pareto has a heavy tail and therefore its tail is a straight line in a log-log scale. The bookstore has a much heavier tail than the online auction site, mainly due to long, robot-initiated sessions [11].

2.4 Function Layer Characterisation

Workload characterization at the function layer includes a breakdown of the number of e-business functions customers invoke during their sessions as well as a global analysis across all users and several time scales. A multi scale analysis shows a twofold increase in bid-submission traffic toward the end of the day [11].

2.5 Request Layer Characterisation

Plotting the number of HTTP requests at several time scales — such as one hour and five seconds — to conduct a visual inspection is a first step to understanding the process by which requests arrive at the site. an apparently strong dependence shows long sequences of increased or decreased volume(trends), particularly at intermediate time scales. We can perform a multiple timescale quantification of the arrival process's dependence on HTTP requests by drawing a variance time plot (VTP) for different time scales. The VTP plot is a log-log plot of the sample variance against the time scale; it helps detect and quantify self-similarity.

The workload component should be at the SUT interface. Each component should represent as homogeneous a group as possible. Combining very different users into a site workload may not be meaningful. Domain of the control affects the component: Example: mail system designer are more interested in determining a typical mail session than a typical user session. Do not use parameters that depend upon the system, e.g., the elapsed time, CPU time. Characteristics of service requests [11]:

1. Arrival Time
2. Type of request or the resource demanded
3. Duration of the request
4. Quantity of the resource demanded, for example, pages of memory
5. Exclude those parameters that have little impact.

2.6 Selection of Workload Components

The workload component should be at the SUT (i.e., system under test) interface [11]

1. Each component should represent as homogeneous a group as possible
   E.g., combining very different users into a site workload may not be meaningful.
2. Purpose of study and domain of control also affect the choice
   E.g., a mail system designer is more interested in a typical mail session than a typical user session involving many applications.

2.7 Selection of Workload Parameters

1. Do not use parameters that depend upon the system.

   E.g., the elapsed time, CPU time [11]

2. Instead, only use parameters that depend on workload itself.

   E.g., characteristics of service requests:

   a. Arrival Time
   b. Type of request or the resource demanded
   c. Duration of the request
   d. Quantity of the resource demanded, for example, buffer space

3. Exclude those parameters that have little impact.

3. PREVIOUS APPRAOCHES IN PUBLIC KEY CRYPTOGRAPHY

3.1 Brief Overview of PKC

One of the weaknesses some point out about symmetric key encryption is that two users attempting to communicate with each other need a secure way to do so; otherwise, an attacker can easily pluck the necessary data from the stream. In November 1976, a paper published in the journal IEEE Transactions on Information Theory, titled "New Directions in Cryptography," addressed this problem and offered up a solution: **public-key encryption** [10].

Also known as **asymmetric-key** encryption, public-key encryption uses two different keys at once -- a combination of a private key and a public key. The private key is known only to your computer, while the public key is given by your computer to any computer that wants to communicate securely with it. To decode an encrypted message, a computer must use the public key, provided by the originating computer, and its own private key. Although a message sent from one computer to another won't be secure since the public key used for encryption is published and available to anyone, anyone who picks it up can't read it without the private key. The key pair is based on prime numbers (numbers that only have divisors of itself and one, such as 2, 3, 5, 7, 11 and so on) of long length. This makes the system extremely secure, because there is essentially an infinite number of prime numbers available, meaning there are nearly infinite possibilities for keys. One very popular public-key encryption program is **Pretty Good Privacy (PGP)**, which allows you to encrypt almost anything.

The sending computer encrypts the document with a symmetric key, then encrypts the symmetric key with the public key of the receiving computer. The receiving computer uses its private key to decode the symmetric key. It then uses the symmetric key to decode the document [10].

To implement public-key encryption on a large scale, such as a secure Web server might need, requires a different approach. This is where **digital certificates** come in. A digital certificate is basically a unique piece of code or a large number that says that the Web server is trusted by an independent source known as a **certificate authority**. The certificate authority acts as a middleman that both computers trust. It confirms that each computer is in fact who it says it is, and then provides the public keys of each computer to the other [14].

3.1.1 Basic Terminology

- **Asymmetric Keys**
  Two related keys, a public key and a private key, that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.
- **Public Key Certificate**
  A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.
- **Public Key Cryptographic Algorithm**
  A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.
- **Public Key Infrastructure (PKI)**
  A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

3.1.2 Distribution of Public Keys

Several Techniques have been proposed for the distribution of public keys. Virtually, all these proposals can be grouped into the following schemes [14]:

1. **Public Announcement**
   The point of public-key encryption is that the public key is public. Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large. For example, because of the growing popularity of PGP, which makes use of RSA, many PGP users have adopted the practice of appending their public key to messages that they send to public forums, such as USENET newsgroups and Internet mailing lists.

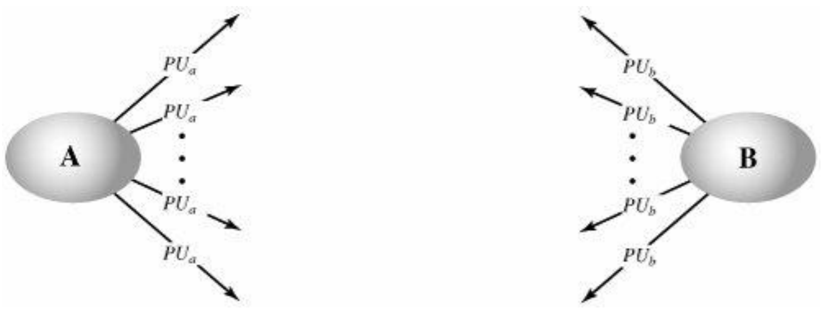


Figure 4 Uncontrolled Public key Distribution

Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement. That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication [14].

2. **Publicly available directory**

   A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys. Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization.

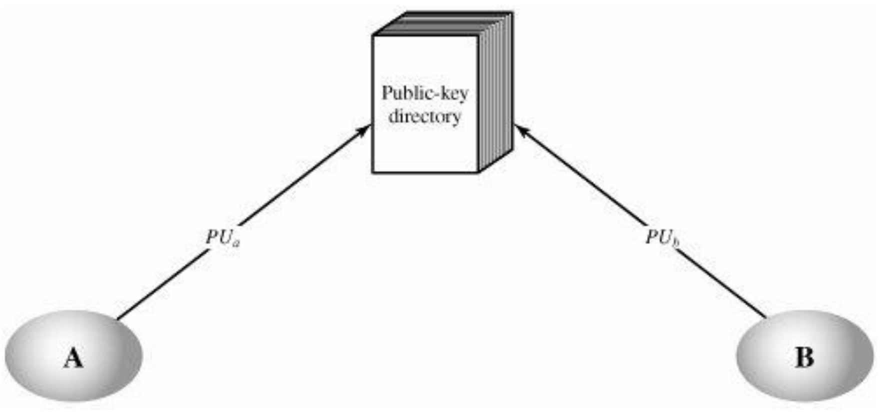


Figure 5 Public Key Publication

This scheme is clearly more secure than individual public announcements but still has vulnerabilities. If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant. Another way to achieve the same end is for the adversary to tamper with the records kept by the authority.

3. **Public-key authority**

   Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. A typical scenario is illustrated in Figure. As before, the scenario assumes that a central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.

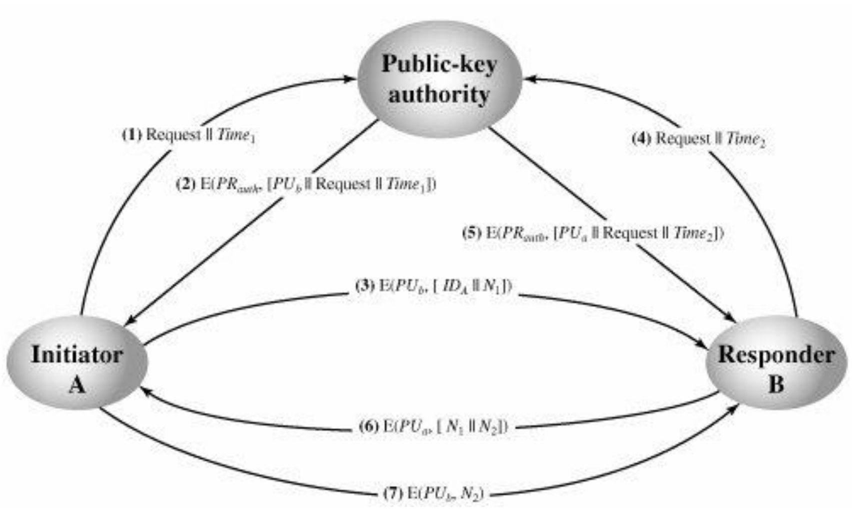

Public-key authority
(1) Request ‖ $Time_1$
(2) E(PR_{auth}, [PU_b ‖ Request ‖ $Time_1$])
(4) Request ‖ $Time_2$
(5) E(PR_{auth}, [PU_a ‖ Request ‖ $Time_2$])
(3) E(PU_b, [ID_A ‖ N_1])
Initiator A
Responder B
(6) E(PU_a, [N_1 ‖ N_2])
(7) E(PU_b, N_2)

Figure 6 Public Key Distribution Scenario

4. **Public Key Certificate**

   The previous scenario is attractive, yet it has some drawbacks. The public-key authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact. As before, the directory of names and public keys maintained by the authority is vulnerable to tampering.

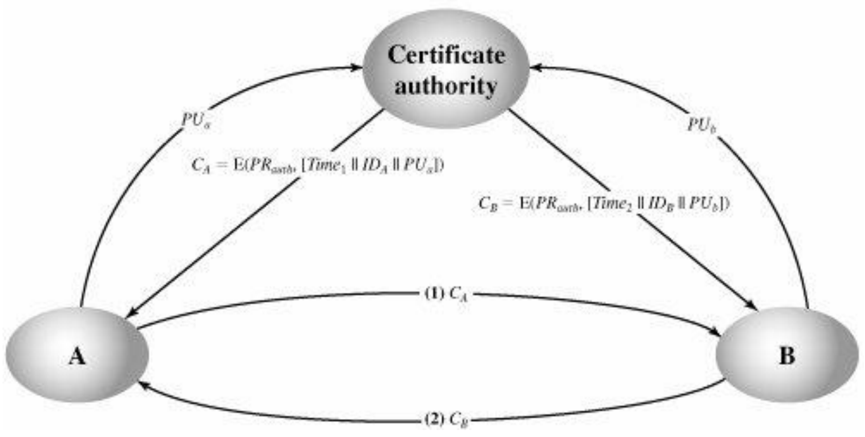

Certificate authority
PU_a
PU_b
$C_A = E(PR_{auth}, [Time_1 \| ID_A \| PU_a])$
$C_B = E(PR_{auth}, [Time_2 \| ID_B \| PU_b])$
(1) C_A
(2) C_B
A
B

Figure 7 Exchange of Public Key Certificates

An alternative approach is to use certificates that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority. In essence, a certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party. Typically, the third party is a certificate authority, such as a government agency or a financial institution that is trusted by the user community. A user can present his or her public key to the authority in a secure manner and obtain a certificate. The user can then publish the certificate. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature. A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority.

3.2 The RSA Algorithm

The pioneering paper by Diffie and Hellman in 1977 introduced a new approach to cryptography and, in effect, challenged cryptologists to come up with a cryptographic algorithm that met the requirements for public-key systems. A number of algorithms have been proposed for public-key cryptography. Some of these, though initially promising, turned out to be breakable.

One of the first successful responses to the challenge was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978. The Rivest-Shamir-Adleman (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption [14].

3.2.1 The Basic Idea

The RSA scheme is a block cipher in which the plaintext and ciphertext are integers between 0 and n - 1 for some n. A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024} [13].

The RSA algorithm — named after Ron Rivest, Adi Shamir, and Leonard Adleman — is based on a property of positive integers that we describe below:

When n is a product of two primes, in arithmetic operations modulo n, the exponents behave modulo the totient φ(n) of n. For example, consider arithmetic modulo 15, since 15 = 3 × 5, for the totient of 15, we have φ(15) = 2 × 4 = 8. You can easily verify the following:

$5^7 \cdot 5^4 \bmod 15 = 5^{(7+4) \bmod 8} \bmod 15 = 5^3 \bmod 15 = 125 \bmod 15 = 5$

$(4^3)^5 \bmod 15 = 4^{(3\times5) \bmod 8} \bmod 15 = 4^7 \bmod 15 = 4$

3.2.2 Key Generation

RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key. The keys for the RSA algorithm are generated the following way [13]:

1. Choose two distinct prime numbers p and q.

For security purposes, the integer p and q should be chosen at random, and should be of similar bit-length. Prime integers can be efficiently found using a primality test.

2. Compute n = pq.

   n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
3. Compute φ(n) = φ(p)φ(q) = (p − 1)(q − 1) = n - (p + q -1), where φ is Euler's totient function.
4. Choose an integer e such that 1 < e < φ(n) and gcd(e, φ(n)) = 1; i.e., e and φ(n) are coprime.

   'e' is released as the public key exponent.

   'e' having a short bit-length and small Hamming weight results in more efficient encryption – most commonly $2^{16} + 1 = 65,537$. However, much smaller values of e (such as 3) have been shown to be less secure in some settings.[5]
5. Determine d as $d \equiv e^{-1}$ (mod φ(n)); i.e., d is the multiplicative inverse of e (modulo φ(n)).

   This is more clearly stated as: solve for d given d·e ≡ 1 (mod φ(n))

   This is often computed using the extended Euclidean algorithm. Using the pseudo code in the Modular integers section, inputs a and n correspond to e and φ(n), respectively.

   d is kept as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and φ(n) must also be kept secret because they can be used to calculate d.

3.2.3 Encryption

Alice transmits her public key (n, e) to Bob and keeps the private key d secret. Bob then wishes to send message M to Alice. He first turns M into an integer m, such that 0≤m<n by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c corresponding to [13]

$$c \equiv m^e \ (mod\ n)$$

This can be done efficiently, even for 500-bit numbers, using Modular exponentiation. Bob then transmits c to Alice. Note that at least nine values of m will yield a ciphertext c equal to m, but this is very unlikely to occur in practice.

3.2.4 Decryption

Alice can recover m from c by using her private key exponent d via computing

$$m \equiv c^d \ (mod\ n)$$

Given m, she can recover the original message M by reversing the padding scheme [9].

3.2.5 An Illustrative Example

Here is an example of RSA encryption and decryption. The parameters used here are artificially small,

1. Choose two distinct prime numbers, such as

$$p = 61 \text{ and } q = 53$$

2. Compute *n* = *pq* giving

$$n = 61 \times 53 = 3233$$

3. Compute the totient of the product as φ(*n*) = (*p* − 1)(*q* − 1) giving

$$\varphi(3233) = (61 - 1)(53 - 1) = 3120$$

4. Choose any number 1 < *e* < 3120 that is coprime to 3120. Choosing a prime number for *e* leaves us only to check that *e* is not a divisor of 3120.

$$\text{Let } e = 17$$

5. Compute *d*, the modular multiplicative inverse of *e* (mod φ(*n*)) yielding,

$$d = 2753$$

   Worked example for the modular multiplicative inverse:

$$e \times d \ \bmod \ \varphi(n) = 1$$

$$17 \times 2753 \ \bmod \ 3120 = 1$$

The public key is (*n* = 3233, *e* = 17). For a padded plaintext message *m*, the encryption function is

$$c(m) = m^{17} \ \bmod \ 3233$$

The private key is (*n* = 3233, *d* = 2753). For an encrypted ciphertext *c*, the decryption function is

$$m(c) = c^{2753} \bmod 3233$$

For instance, in order to encrypt *m* = 65, we calculate

$$c = 65^{17} \bmod 3233 = 2790$$

To decrypt *c* = 2790, we calculate

$$m = 2790^{2753} \bmod 3233 = 65$$

Both of these calculations can be computed efficiently using the square-and-multiply algorithm for modular exponentiation. In real-life situations the primes selected would be much larger; in our example it would be trivial to factor *n*, 3233 (obtained from the freely available public key) back to the primes *p* and *q*. Given *e*, also from the public key, we could then compute *d* and so acquire the private key.

3.2.6 Proof of Correctness

The proof of the correctness of RSA is based on Fermat's little theorem [13]. This theorem states that if *p* is prime and *p* does not divide an integer *a* then

$$a^{p-1} \equiv 1 \pmod{p}$$

We want to show that $m^{ed} \equiv m \pmod{pq}$ for every integer *m* when *p* and *q* are distinct prime numbers and *e* and *d* are positive integers satisfying

$$ed \equiv 1 \pmod{(p-1)(q-1)}.$$

We can write

$$ed - 1 = h(p-1)(q-1)$$

for some nonnegative integer *h*.

To check two numbers, like m^{ed} and *m*, are congruent mod *pq* it suffices (and in fact is equivalent) to check they are congruent mod *p* and mod *q* separately. To show $m^{ed} \equiv m \pmod{p}$, we consider two cases: $m \equiv 0 \pmod{p}$ and $m \not\equiv 0 \pmod{p}$.

In the first case m^{ed} is a multiple of *p*, so $m^{ed} \equiv 0 \equiv m \pmod{p}$. In the second case

$$m^{ed} = m^{(ed-1)}m = m^{h(p-1)(q-1)}m = (m^{p-1})^{h(q-1)}m \equiv 1^{h(q-1)}m \equiv m \ (mod\ p)$$

where we used Fermat's little theorem to replace m^{p-1} mod p with 1.

The verification that $m^{ed} \equiv m$ (mod q) proceeds in a similar way, treating separately the cases $m \equiv 0$ (mod q) and $m \not\equiv 0$ (mod q), using Fermat's little theorem for modulus q in the second case.

This completes the proof that, for any integer m,

$$(m^e)^d \equiv m \pmod{pq}$$

3.2.7 Cryptanalysis of RSA

While cryptography is the science concerned with the design of ciphers, cryptanalysis is the related study of breaking ciphers. Cryptography and cryptanalysis are somehow complimentary sciences: development in one is usually followed by further development in the other. In particular, cryptanalysis is an important tool for vulnerability assessment of cryptosystems [2].

In an encryption scheme, the main objective of the adversary is to recover the plaintext M from the related ciphertext. If he is successful, we say he has broken the system. In the case of digital signatures, the goal of the adversary is to forge signatures. A more ambitious attack is to recover the private key d. If achieved, the adversary can now decrypt all ciphertexts and forge signatures at will. In this case the only solution is the revocation of the key.

As it is already standard in examples of use of cryptography, we name three entities involved in the system as Alice, Bob and Caroline. Alice and Bob wish to securely communicate with each other, while Caroline is a malicious adversary, trying passively or actively to disturb the communication.

The problem of finding non-trivial factors of a given positive composite integer n is known as the integer factorization problem. The integer factorization problem is widely believed to be a hard problem, i.e., there seems to be no polynomial time algorithm that solves the problem for a large proportion of possible inputs. Note that factoring is obviously not always hard, but a hard instance of the problem can be easily created, by simply multiplying two large chosen prime numbers. That is exactly what we do when working with RSA.

As we have seen, the security of the RSA cryptosystem is intimately related to the integer factorization problem. If an adversary can factor the modulus n, he can efficiently calculate the private exponent. In this case we say that he has completely broken the encryption scheme: he not only can recover a particular plaintext M, but also decrypt all ciphertexts encrypted with the respective public key.

Therefore, although factoring methods are not used in practice in attacks against RSA, they are important in deriving lower bounds for key sizes and properties of the security parameters. Taking into account the value of data and the threat model, parameters should be chosen such that factoring the modulus is computationally infeasible.

Factoring methods can be divided into special-purpose and general-purpose factoring methods. Special purpose methods depend on special properties of the number to be factored, as the size of the smallest factor p of n, the factorization of p – 1, etc. General-purpose methods depend solely on the size of n.

3.3 Diffie-Hellman key exchange

Diffie–Hellman key exchange (D–H) is a specific method of securely exchanging cryptographic keys over a public channel and was the first specific example of public-key cryptography as originally conceptualized by Ralph Merkle. D–H is one of the earliest practical examples of public key exchange implemented within the field of cryptography. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher [3].

The scheme was first published by Whitfield Diffie and Martin Hellman in 1976. By 1975, James H. Ellis, Clifford Cocks and Malcolm J. Williamson within GCHQ, the British signals intelligence agency, had also shown how public-key cryptography could be achieved; although, their work was kept secret until 1997.

Although Diffie–Hellman key agreement itself is an anonymous (non-authenticated) key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provide perfect forward secrecy in Transport Layer Security's ephemeral modes (referred to as EDH or DHE depending on the cipher suite).

3.3.1 Cryptographic explanation

The simplest and the original implementation of the protocol uses the multiplicative group of integers modulo p, where p is prime, and g is a primitive root modulo p. Here is an example of the protocol, with non-secret values in blue, and secret values in red.

1. Alice and Bob agree to use a prime number p = 23 and base g = 5 (which is a primitive root modulo 23).
2. Alice chooses a secret integer a = 6, then sends Bob A = g^a mod p
   - A = 5^6 mod 23 = 8
3. Bob chooses a secret integer b = 15, then sends Alice B = g^b mod p
   - B = 5^{15} mod 23 = 19
4. Alice computes s = B^a mod p
   - s = 19^6 mod 23 = 2
5. Bob computes s = A^b mod p
   - s = 8^{15} mod 23 = 2
6. Alice and Bob now share a secret (the number 2).

Both Alice and Bob have arrived at the same value, because $(g^a)^b$ (for Bob, 8^{15} mod 23 = $(g^a \bmod p)^b \bmod p = (g^a)^b \bmod p$) and $(g^b)^a$ are equal mod *p*. Note that only *a*, *b*, and $(g^{ab} \bmod p = g^{ba} \bmod p)$ are kept secret. All the other values – *p*, *g*, g^a mod *p*, and g^b mod *p* – are sent in the clear. Once Alice and Bob compute the shared secret they can use it as an encryption key, known only to them, for sending messages across the same open communications channel.

Of course, much larger values of *a*, *b*, and *p* would be needed to make this example secure, since there are only 23 possible results of *n* mod 23. However, if *p* is a prime of at least 300 digits, and *a* and *b* are at least 100 digits long, then even the fastest modern computers cannot find *a* given only *g*, *p*, g^b mod *p* and g^a mod *p*. The problem such a computer needs to solve is called the discrete logarithm problem. The computation of g^a mod *p* is known as modular exponentiation and can be done efficiently even for large numbers. Note that *g* need not be large at all, and in practice is usually a small prime (like 2, 3, 5...) because primitive roots usually are quite numerous.

4. ELLIPTIC CURVE CRYPTOGRAPHY – PROPOSED APPROACH

Over the past 30 years, public key cryptography has become a mainstay for secure communications over the Internet and throughout many other forms of communications. It provides the foundation for both key management and digital signatures. In key management, public key cryptography is used to distribute the secret keys used in other cryptographic algorithms (e.g. DES). For digital signatures, public key cryptography is used to authenticate the origin of data and protect the integrity of that data. For the past 20 years, Internet communications have been secured by the first generation of public key cryptographic algorithms developed in the mid-70s. Notably, they form the basis for key management and authentication for IP encryption (IKE/IPSEC), web traffic (SSL/TLS) and secure electronic mail [5].

Over the last twenty years however, new techniques have been developed which offer both better performance and higher security than these first generation public key techniques. The best assured group of new public key techniques is built on the arithmetic of elliptic curves [6].

4.1 A Brief Overview of ECC

Elliptic Curve Cryptography (ECC) was discovered in 1985 by Victor Miller (IBM) and Neil Koblitz (University of Washington) as an alternative mechanism for implementing public-key cryptography. Public-key algorithms create a mechanism for sharing keys among large numbers of participants or entities in a complex information system. Unlike other popular algorithms such as RSA, ECC is based on discrete logarithms that is much more difficult to challenge at equivalent key lengths [7][8].

4.1.1 Advantages of ECC

There are certain unique properties of elliptic curves which made them resilient against the types of attacks that were successful against integer-based algorithms. Therefore an elliptic-curve algorithm (e.g. Elliptic-curve Digital Signature Algorithm = EC-DSA) could have the same level of security of a similar integer-based algorithm (e.g. Digital Signature Algorithm = DSA) using much fewer key bits. Table 2 shows the number of key bits necessary to have equivalent levels of security for integer-based algorithms versus elliptic-curve algorithms [7].

Table 2 Key Length (in bits) for equivalent security

Integer Algorithm (e.g. DSA)	**Elliptic Curve algorithm (e.g. EC-DSA)**
512	106
768	132
1024	160
2048	210
21000	600

For the most commonly used 1024-bit keys for an integer-based algorithm, the elliptic-curve counterpart only requires 160-bit keys for the equivalent security. This is a 7x reduction in the space required to store these keys, or a similar reduction in bandwidth required to transmit these keys over a wireless network. Furthermore, this reduction in the size of data objects allows much faster completion of the algorithms [7].

Table 3 A Comparison of Public key cryptosystems

Type of Cryptosystem	**Examples**	**Complexity of best known Attack algorithm [8]**
Integer-based Cryptosystem	RSA, Diffie-Hellman	$\exp[1.923(\log n)^{1/3}(\log \log n)^{2/3}]$ (Sub-exponential)
Elliptic Curve Cryptosystem	ECDH, ECDSA	$\sqrt{n}$ (Fully exponential)

4.2 Elliptic Curves : Better Trapdoor

An elliptic curve is the set of points that satisfy a specific mathematical equation. The equation for an elliptic curve looks something like this [6]:

$$y^2 = x^3 + ax + b$$

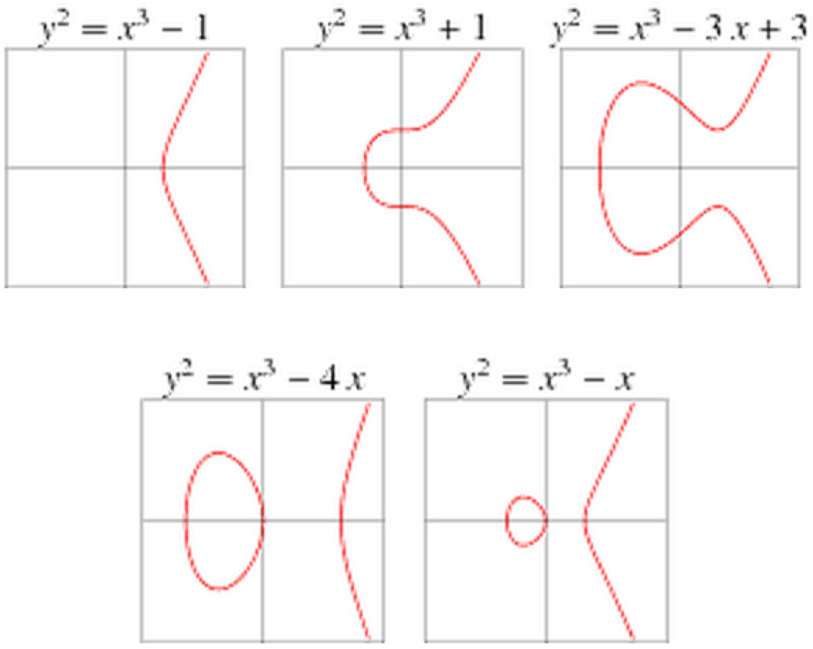


Figure 8 Some Examples of Elliptic Curves

These curves are of great use in a number of applications, largely because it possible to take two points on such a curve and generate a third. In fact, we can show that by defining an addition operation and introducing an extra point, 1, the points on an elliptic curve form an additive abelian group.

Such a group can then be used to create an analogue of the discrete logarithm problem which is the basis for several public key cryptosystems.

4.2.1 Properties of Elliptic Curves

The Elliptic Curve obeys following properties [6][14]:

- **Closure**

  If P and Q are on elliptic curve, then (P+Q) will also lie on the same curve.

- **Commutativity**

$$P + Q = Q + P$$

- **Associativity**

$$(P + Q) + R = P + (Q + R)$$

- **Existence of identity element**

  There exists an element ‘O’, such that

$$P + O = O + P = P$$

- **Existence of inverses**

  There exists an element ‘–P’, such that

$$(-P) + P = P + (-P) = O$$

4.2.2 Addition on Elliptic Curves

The rules for addition can be stated as follows: If three points on an elliptic curve lie on a straight line, their sum is O. From this definition, we can define the rules of addition over an elliptic curve [6][14].

1. O serves as the additive identity. Thus, $O = -O$; for any point P on the elliptic curve, $P + O = P$. In what follows, we assume $P \neq O$ and $Q \neq O$.
2. The negative of a point P is the point with the same x coordinate but the negative of the y coordinate; that is, if $P = (x, y)$, then $-P = (x, -y)$. Note that these two points can be joined by a vertical line. Note that $P + (-P) = P - P = O$.
3. To add two points P and Q with different x coordinates, draw a straight line between them and find the third point of intersection R. It is easily seen that there is a unique point R that is the point of intersection (unless the line is tangent to the curve at either P or Q, in which case we take $R = P$ or $R = Q$, respectively). To form a group structure, we need to define addition on these three points: $P + Q = -R$. That is, we define $P + Q$ to be the mirror image (with respect to the axis) of the third point of intersection.
4. The geometric interpretation of the preceding item also applies to two points, P and $-P$, with the same coordinate. The points are joined by a vertical line, which can be viewed as also intersecting the curve at the infinity point. We therefore have $P + (-P) = O$, which is consistent with item.
5. To double a point Q, draw the tangent line and find the other point of intersection S. Then $Q + Q = 2Q = -S$.

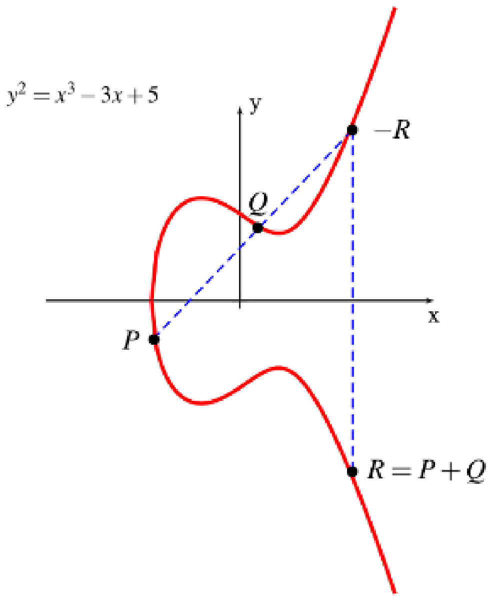


Figure 9 Point Addition on Elliptic Curve

Algorithm for Point Addition [7]:

INPUT:

Elliptic curve points $P = (x_1, y_1)$ *and* $Q = (x_2, y_2),\ P \neq Q$

OUTPUT: $R = P + Q = (x_3, y_3)$

1. Compute $\theta = \frac{y_2 + y_1}{x_2 + x_1}$
2. Compute $x_3 = \theta^2 + \theta + x_1 + x_2 + a$
3. Compute $y_3 = \theta(x_1 + x_3) + x_3 + y_1$
4. Return (x_3, y_3)

4.2.3 Point Doubling

To add a point P to itself, a tangent line to the curve is drawn at the point P. If yP is not 0, then the tangent line intersects the elliptic curve at exactly one other point, -R. -R is reflected in the x-axis to R. This operation is called doubling the point P; the law for doubling a point on an elliptic curve group is defined by [4]:

$$P + P = 2P = R$$

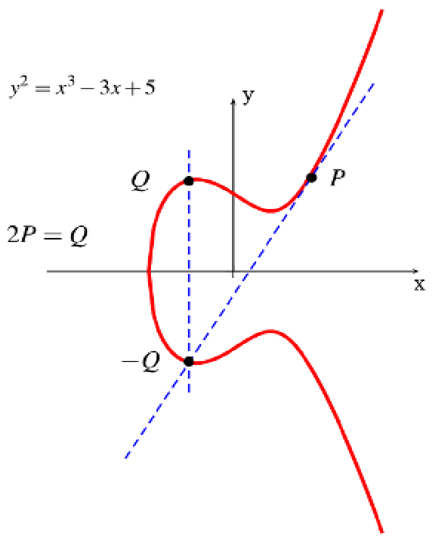


Figure 10 Point Doubling on Elliptic Curve

Algorithm for Point Doubling [7]:

INPUT: Elliptic Curve point $P = (x, y)$

OUTPUT: $R = P + P = (x_3, y_3)$

1. Compute $\theta = x + \frac{y}{x}$
2. Compute $x_3 = \theta^2 + \theta + a$
3. Compute $y_3 = x^2 + (\theta + 1)x_3$
4. Return (x_3, y_3)

4.3 Key Generation

Key generation is an important part where we have to generate both public key and private key. The sender will be encrypting the message with receiver's public key and the receiver will decrypt using his private key.

Now, we have to select a number d within the range of 'n'. Using the following equation, we can generate the public key [1]

$$Q = d * P$$

Where,

d => The random number that we have selected within the range (1 to n-1) (Private key).

P => Point on the curve.

Q => Public key

4.4 Representing Plaintext

- Let $E\colon\ y^2 \equiv x^3 + bx + c(mod\, p)$
- Message m (represented as a number) will be embedded in the x-coordinate of a point [1].
- Adjoin a few bits at the end of m and adjust until we get a number x such that $x^3 + bx + c$ is square mod p.

4.4.1 Illustration

- Let $p = 179$ and $E\colon\ y^2 = x^3 + 2x + 7$
- If failure rate of $\frac{1}{2^{10}}$, then we may take $K = 10$.
- We need $m.K + K < 179$, we need $0 \leq m \leq 16$
- Suppose our message is $m = 5$. We consider x of the form

$$m.K + j = 50 + j$$

- The possible choices for x are $50,\ 51, \dots,\ 59$.
  - For $x = 51$, we get

$$x^3 + 2x + 7 \equiv 121\big(mod\ 179\big) \qquad 11^2 = 121(mod\ 179)$$

- Thus, we represent the message $m = 5$ by the point

$$P_m = (51,\ 11)$$

- The message m can be recovered by $m = \left\lfloor \frac{51}{10} \right\rfloor = 5$

4.5 Encryption

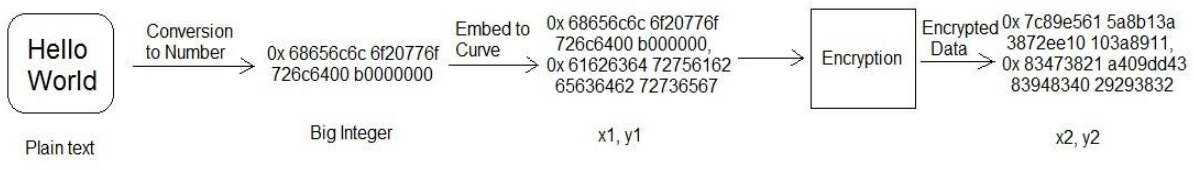

Hello World
Plain text
Conversion to Number
0x 68656c6c 6f20776f 726c6400 b0000000
Big Integer
Embed to Curve
0x 68656c6c 6f20776f 726c6400 b000000, 0x 61626364 72756162 65636462 72736567
x1, y1
Encryption
Encrypted Data
0x 7c89e561 5a8b13a 3872ee10 103a8911, 0x 83473821 a409dd43 83948340 29293832
x2, y2

Figure 11 Encryption Process

- Let 'm' be the message that we are sending. We have to represent this message on the curve. Consider *'m'* has the point *'M'* on the curve *'E'*.
- Randomly select 'k' from 1 to (n-1) [14].
- Two cipher texts will be generated let it be C1 and C2.

$$C1 = k * P$$
$$C2 = M + k * Q$$

C1 and C2 will be sent.

- Here multiplication of a scalar with a point on curve $(k * P)$ will be computed as follows,

$$11P = \big((P * 2) * 2 + P\big) * 2 + P$$
$$9P = \Big(\big((P * 2) * 2\big) * 2\Big) + P$$

4.6 Decryption

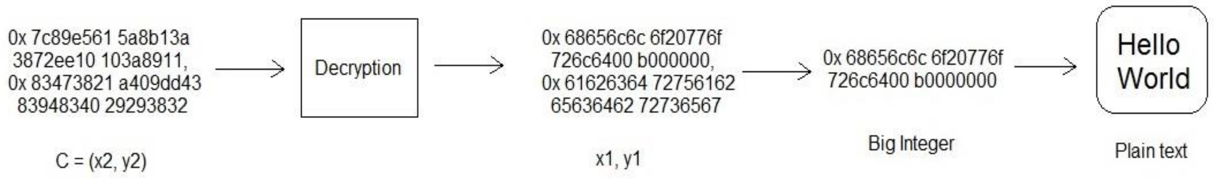

0x 7c89e561 5a8b13a 3872ee10 103a8911, 0x 83473821 a409dd43 83948340 29293832
C = (x2, y2)
Decryption
0x 68656c6c 6f20776f 726c6400 b000000, 0x 61626364 72756162 65636462 72736567
x1, y1
0x 68656c6c 6f20776f 726c6400 b0000000
Big Integer
Hello World
Plain text

Figure 12 Decryption Process

- We have to get back the message ‘m’ that was send to us,

$$M = C2 - d * C1$$

- M is the point representing original message that we have sent [14].

4.7 Proof of Correctness of ECC

- Message can be retrieved as follows,

$$M = C2 - d * C1$$

i.e. ‘M’ can be represented as ‘C2 – d * C1′

- Here,

$$C2 = M + k * Q$$

$$C1 = k * P$$

- Therefore,

$$\begin{aligned} C2 - d * C1 &= (M + k * Q) - d * (k * P) \\ &= M + k * d * P - d * k * P \\ &= M\ (Original\ Message) \end{aligned}$$

5. IMPLEMENTATION AND SCREENSHOTS

5.1 Elliptic Curve Generation

5.1.1 Construction of Elliptic Curve

Constructs an elliptic curve over the finite field of 'mod' elements.

- The equation of the curve is on the form: $y^2 = x^3 + ax + b$.
- p must a prime number

Code Snippet:

```
public EllipticCurve(BigInteger a, BigInteger b, BigInteger p) throws
InsecureCurveException {
        this.a = a;
        this.b = b;
        this.p = p;
        if (!p.isProbablePrime(PRIMESECURITY)) {        }
        if (isSingular()) throw new
InsecureCurveException(InsecureCurveException.SINGULAR, this);
        byte[] pb = p.toByteArray();
        if(pb[0] == 0) pointcmpsize = pb.length;
        else pointcmpsize = pb.length + 1;
        name = "";
   }
   public EllipticCurve(ECParameters ecp) throws InsecureCurveException {
        this(ecp.a(),ecp.b(),ecp.p());
        order = ecp.order();
        name = ecp.toString();
        try{
            generator = new ECPoint(this, ecp.generatorX(), ecp.generatorY());
            generator.fastCache();
        }
        catch (NotOnMotherException e){
            System.out.println("Error defining EllipticCurve: generator not on
mother!");
```

```
        }
  }
```

5.1.2 Checking Whether a point is on the Elliptic Curve

This functions checks whether a point is on the Elliptic Curve.

- Returns *true* if point is on the Curve
- Returns *false* if point is not on the Curve

Code Snippet:

```
public boolean onCurve(ECPoint q){
        if (q.isZero()) return true;
        BigInteger y_square = (q.gety()).modPow(new BigInteger("2"),p);
        BigInteger x_cube = (q.getx()).modPow(new BigInteger("3"),p);
        BigInteger x = q.getx();
        BigInteger dum = ((x_cube.add(a.multiply(x))).add(b)).mod(p);
        if (y_square.compareTo(dum) == 0) return true;
        else return false;
  }
```

5.2 Elliptic Curve Cryptosystem

5.2.1 Encryption

Using the key, encrypting the plaintext from input file.

Code Snippet:

```
  public byte[] encrypt(byte[] input,int numbytes, Key key) {
        ECKey ek = (ECKey) key;
        byte[] res=new byte[ek.mother.getPCS()+numbytes];
        hash.reset();
        BigInteger rk = new BigInteger(ek.mother.getp().bitLength() + 17, Rand.om);
        if (ek.mother.getOrder() != null) {
                rk = rk.mod(ek.mother.getOrder());
        }
        ECPoint gamma = ek.mother.getGenerator().multiply(rk);
        ECPoint sec = ek.beta.multiply(rk);
```

```
            System.arraycopy(gamma.compress(),0,res,0,ek.mother.getPCS());
            hash.update(sec.getx().toByteArray());
            hash.update(sec.gety().toByteArray());
            byte[] digest = hash.digest();
            for(int j = 0; j < numbytes; j++) {
                res[j+ek.mother.getPCS()]=(byte) (input[j]^digest[j]);
            }
            return res;
    }
```

5.2.2 Decryption

Generating Plaintext from ciphertext using the key.

Code Snippet:

```
    public byte[] decrypt(byte[] input, Key key) {
            ECKey dk = (ECKey) key;
            byte[] res=new byte[input.length-dk.mother.getPCS()];
            byte[] gammacom=new byte[dk.mother.getPCS()];
            hash.reset();
            System.arraycopy(input,0,gammacom,0,dk.mother.getPCS());
            ECPoint gamma = new ECPoint(gammacom,dk.mother);
            ECPoint sec = gamma.multiply(dk.sk);
            if(sec.isZero()) {
                hash.update(BigInteger.ZERO.toByteArray());
                hash.update(BigInteger.ZERO.toByteArray());
            } else {
                hash.update(sec.getx().toByteArray());
                hash.update(sec.gety().toByteArray());
            }
            byte[] digest = hash.digest();
            for(int j = 0; j < input.length-dk.mother.getPCS(); j++) {
                res[j]=(byte) (input[j+dk.mother.getPCS()]^digest[j]);
            }
```

```
        return res;
    }
```

5.3 Implementation of Elliptic Curve Point Algorithms

5.3.1 Point Addition

Adds another elliptic curve point to this point. Returns the sum of *this* point with point ‘q’.

Code Snippet:

```
    public ECPoint add(ECPoint q) throws NoCommonMotherException{
        if (!hasCommonMother(q)) throw new NoCommonMotherException();
        if (this.iszero) return q;
        else if (q.isZero()) return this;
        BigInteger y1 = y;
        BigInteger y2 = q.gety();
        BigInteger x1 = x;
        BigInteger x2 = q.getx();
        BigInteger alpha;
        if (x2.compareTo(x1) == 0) {
            if (!(y2.compareTo(y1) == 0)) return new ECPoint(mother);
            else {
                alpha =
((x1.modPow(TWO,mother.getp())).multiply(THREE)).add(mother.geta());
                alpha =
(alpha.multiply((TWO.multiply(y1)).modInverse(mother.getp()))).mod(mother.getp()
);
            }

        } else {
            alpha =
((y2.subtract(y1)).multiply((x2.subtract(x1)).modInverse(mother.getp()))).mod(mothe
r.getp());
        }
        BigInteger x3,y3;
        x3 =
(((alpha.modPow(TWO,mother.getp())).subtract(x2)).subtract(x1)).mod(mother.getp(
));
        y3 = ((alpha.multiply(x1.subtract(x3))).subtract(y1)).mod(mother.getp());
        try{ return new ECPoint(mother,x3,y3); }
        catch (NotOnMotherException e){
            System.out.println("Error in add!!! Result not on mother!");
            return null;
}
```

5.3.2 Point Multiplication

Multiplies a number *coef* to *this* point. Returns the product of *this* point with the number.

Code Snippet:

```
public ECPoint multiply(BigInteger coef) {
        try{
            ECPoint result = new ECPoint(mother);
            byte[] coefb = coef.toByteArray();
            if(fastcache != null) {
                for(int i = 0; i < coefb.length; i++) {
                    result = result.times256().add(fastcache[coefb[i]&255]);
                }
                return result;
            }
            if(cache == null) {
                cache = new ECPoint[16];
                cache[0] = new ECPoint(mother);
                for(int i = 1; i < cache.length; i++) {
                    cache[i] = cache[i-1].add(this);
                }
            }
            for(int i = 0; i < coefb.length; i++) {
                result = 
result.times16().add(cache[(coefb[i]>>4)&15]).times16().add(cache[coefb[i]&15]);
            }
            return result;
        } catch (NoCommonMotherException e) {
            System.out.println("Error in pow!!!");
            return null;
        }
    }
```

5.4 Key Generation

Generates a random secret key (contains also the public key)

Code Snippet:

```
public ECKey(EllipticCurve ec) {
        mother = ec;
        secret = true;
        sk=new BigInteger(ec.getp().bitLength() + 17,Rand.om);
        if (mother.getOrder() != null) sk=sk.mod(mother.getOrder());
        beta=(mother.getGenerator()).multiply(sk);
        beta.fastCache();
}
```

5.5 Snapshots

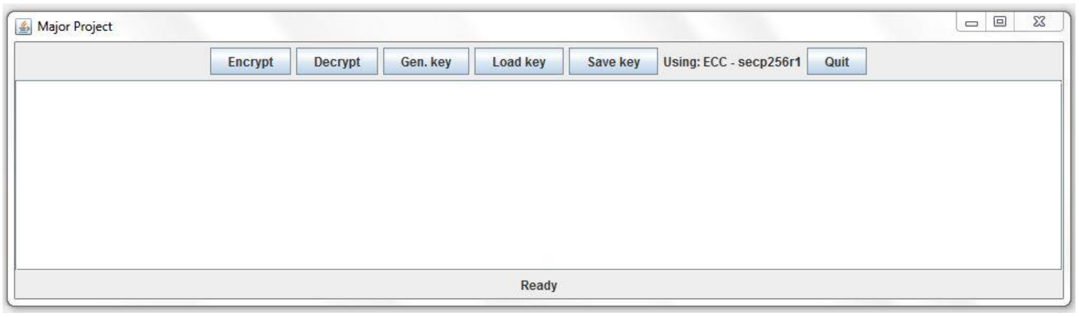
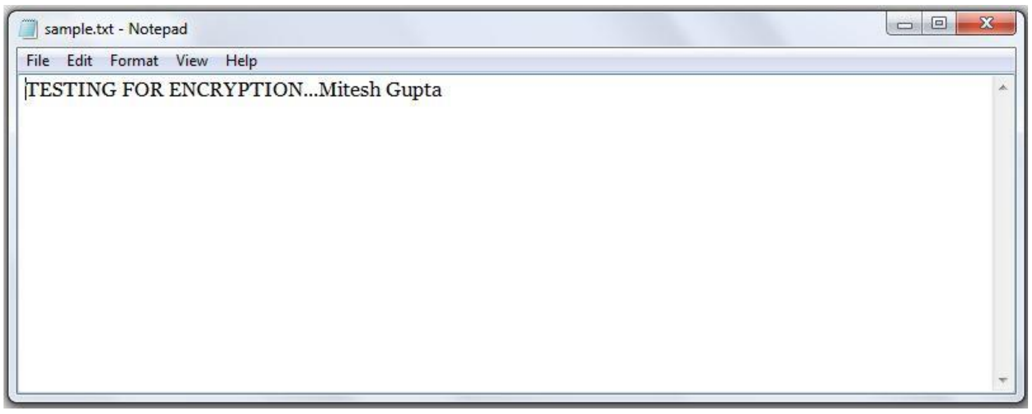
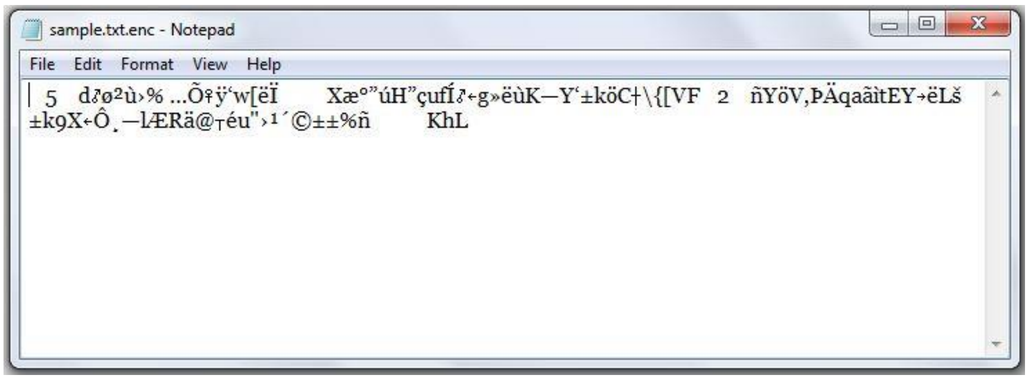


Figure 13 User Interface



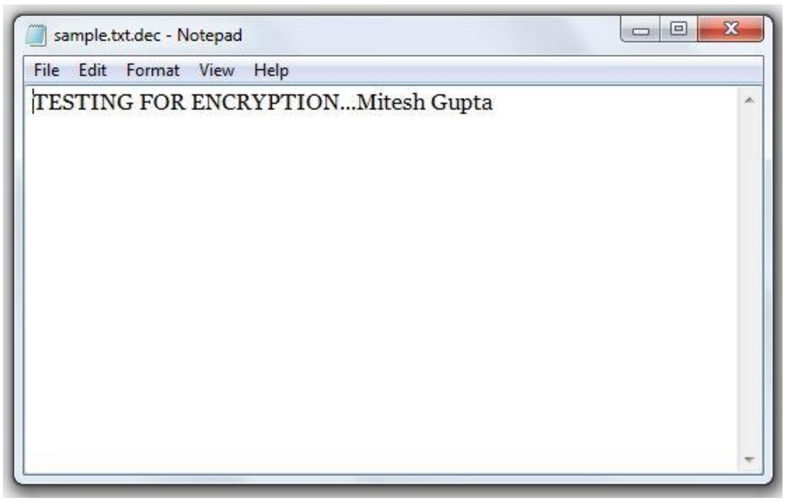
TESTING FOR ENCRYPTION...Mitesh Gupta

Figure 14 Input File for Encryption



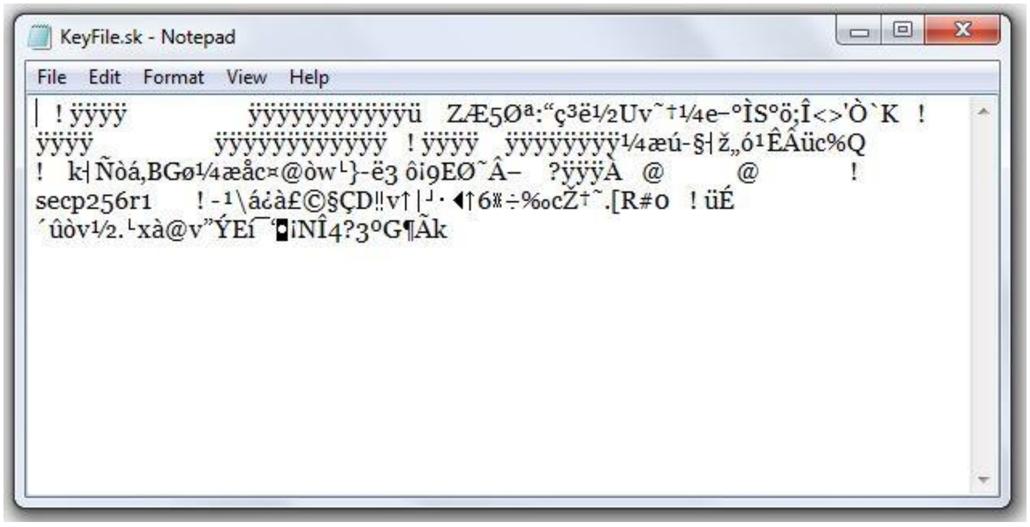
```
 5  d♪ø²ù›% …Õ♀ÿ‘w[ëÏ     Xæ°”úH”çufÍ♪←g»ëùK—Y‘±köC┼\{[VF  2  ñYöV,ÞÄqaãìtEY→ëLš
±k9X←Ô˛—lÆRä@┬éu"›¹´©±±%ñ      KhL
```

Figure 15 Encrypted File after performing Elliptic Curve Encryption



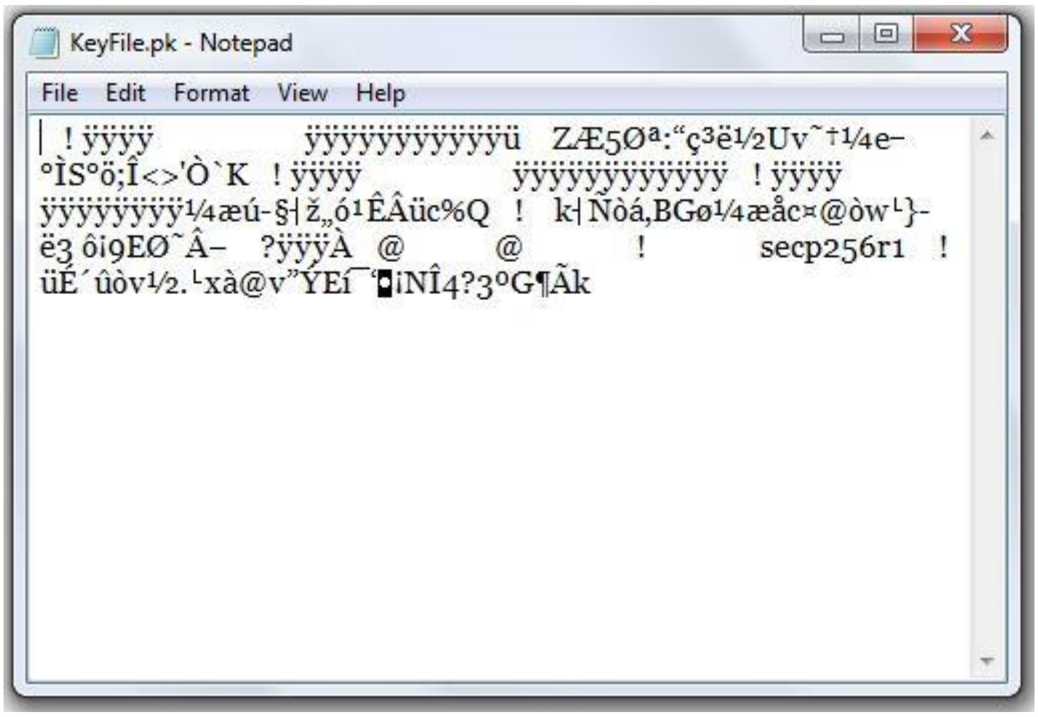
```
TESTING FOR ENCRYPTION...Mitesh Gupta
```

Figure 16 Decrypted File after Elliptic Curve Decryption



```
 ! ÿÿÿÿ          ÿÿÿÿÿÿÿÿÿÿÿÿü  ZÆ5Øª:“ç³ë½Uv˜†¼e–°ÌS°ö;Î<>'Ò`K  !
ÿÿÿÿ          ÿÿÿÿÿÿÿÿÿÿÿÿ  ! ÿÿÿÿ  ÿÿÿÿÿÿÿÿ¼æú-§┤ž„ó¹ÊÁüc%Q
!  k┤Ñòá,BGø¼æåc¤@òw└}-ë3 ôi9EØ˜Â–  ?ÿÿÿÀ  @      @        !
secp256r1   ! -¹\á¿à£©§ÇD‼v↑|┘· ◄↑6‡÷‰cŽ†˜.[R#o  ! üÉ
´ûòv½.└xà@v”ÝEí¯‘◘iNÎ4?3ºG¶Ãk
```

Figure 17 Secret Key for ECC



```
 ! ÿÿÿÿ            ÿÿÿÿÿÿÿÿÿÿÿÿü   ZÆ5Øª:"ç³ë½Uv˜†¼e–
°ÌS°ö;Î<>'Ò`K  ! ÿÿÿÿ           ÿÿÿÿÿÿÿÿÿÿÿÿ  ! ÿÿÿÿ
ÿÿÿÿÿÿÿÿ¼æú-§┤ž„ó¹ÊÂüc%Q  !   k┤Ñòá,BGø¼æåc¤@òw└}-
ë3 ôi9EØ˜Â–   ?ÿÿÿÀ  @        @          !        secp256r1  !
üÉ´ûòv½.└xà@v"ÝEí¯'◘iNÎ4?3ºG¶Ãk
```

Figure 18 Public Key for ECC

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

The above report shows: (1) a study of cryptography algorithms suitable for constrained environments, (2) an understanding of the operations used by Elliptic Curve Cryptography algorithms, and (3) an implementation of the ECC algorithm.

Also described the arithmetic algorithm used to implement elliptic-curve operations. We show that this powerful and mathematically complicated algorithm can be implemented very efficiently using a simple java implementation.

6.2 Future Work

For future work, I plan to expand the algorithm set to include other public-key cryptography algorithms.

Also expand the result set to include the comparative analysis on the basis of parameters like efficiency, security, memory space used, computational time, etc.

REFERENCES

[1] *A (relatively easy to understand) primer on elliptic curve cryptography*. (2014, December 14). Retrieved from ArsTechnica Website: http://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/

[2] *Cryptanalysis of RSA: A Survey.* (2014, December 14). Retrieved from SANS Institute Website: http://www.sans.org/reading-room/whitepapers/vpns/cryptanalysis-rsa-survey-1006

[3] *Diffie–Hellman Key Exchange*. (2014, December 14). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

[4] *Doubling the point P*. (2014, December 14). Retrieved from Certicom Corp. Website: https://www.certicom.com/index.php/213-doubling-the-point-p

[5] *Elliptic Curve Cryptography*. (2014, December 14). Retrieved from Certicom Corp. Website: https://www.certicom.com/ecc

[6] England, M. (2006). Elliptic curve cryptography. *M. Sc Applied Mathematical Science*. Summer: Heriot-Watt University.

[7] Fiskaran, A. M., & Lee, R. B. (2002). Workload Characterization of Elliptic Curve Cryptography and other Network Security Algorithms for Constrained Environments. *5th IEEE Annual Workshop on Workload Characterization (WWC-5).* Austin, Texas, USA: IEEE.

[8] Gupta, V., Stebila, D., Fung, S., Shantz, S. C., Gura, N., & Eberle, H. (2004). Speeding up Secure Web Transactions Using Elliptic Curve Cryptography. *Network and Systems Security Symposium*, (pp. 231--239).

[9] Kak, A. (2014, April 23). *Public Key Cryptography and the RSA Algorithm.* Retrieved from Purdue University Website: https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture12.pdf

[10] Kesssler, G. C. (2010, June 30). An Overview of Cryptography.

[11] Menasce, D. A. (2003). Workload Characterization. *IEEE* . VA, USA: IEEE.

[12] *Public Key Encrytion - HowStuffWorks*. (2014, December 14). Retrieved from HowStuffWorks.com: http://computer.howstuffworks.com/encryption3.htm

[13] *RSA (Cryptosystem)*. (2014, December 14). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/RSA_(cryptosystem)

[14] Stallings, W. (2011). *Cryptography and Network Security - Principles and Practice, Fifth Edition.* Pearson Education, Inc.

[15] *The Case for Elliptic Curve Cryptography - NSA/CSS*. (2014, December 14). Retrieved from NSA/CSS Website: https://www.nsa.gov/business/programs/elliptic_curve.shtml