A Major Project Report On

FEDERAL IDENTITY MANAGEMENT USING HIERARCHICAL IDENTITY-BASED CRYPTOGRAPHY (HIBC)

Submitted in partial fulfillment of the requirements

For the award of the degree of

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

By

AARTI GOEL

(Roll No. 2K12/CSE/01)

Under the guidance of

Associate Prof. Manoj Kumar

Department of COE

Delhi Technological University, Delhi

Department of Computer Engineering

Delhi Technological University, Delhi

2012 – 2014

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person for material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or the other institute of higher learning, except where due acknowledgment has been made in the text.

Signature:

Name: *Aarti Goel*

Roll No: *2K12/CSE/01*

Date:

CERTIFICATE

This is to certify that the project report entitled is a bona fide record of work carried out by AARTI GOEL (2K12/CSE/01) under my guidance and supervision, during the academic session 2012-2014 in partial fulfillment of the requirement for the degree of Master of Technology in Computer Science Engineering from Delhi Technological University, Delhi.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Manoj Kumar
Associate Professor
Department of Computer Engineering
Delhi Technological University
Delhi

ACKNOWLEDGEMENT

With due regards, I hereby take this opportunity to acknowledge a lot of people who have supported me with their words and deeds in completion of my research work as part of this course of Master of Technology in Computer Science Engineering.

To start with I would like to thank the almighty for being with me in each and every step of my life. Next, I thank my parents and family for their encouragement and persistent support.

I would like to express my deepest sense of gratitude and indebtedness to my guide and motivator, **Manoj Kumar**, Associate Professor, Department of Computer Engineering, Delhi Technological University for her valuable guidance and support in all the phases from conceptualization to final completion of the project.

I wish to convey my sincere gratitude to **Dr. O.P. Verma,** Head of Department, and all the faculties and PhD. Scholars of Computer Engineering Department, Delhi Technological University who have enlightened me during my project.

I humbly extend my grateful appreciation to my friends whose moral support made this project possible.

Last but not the least; I would like to thank all the people directly and indirectly involved in successfully completion of this project.

Aarti Goel

Roll No. 2K12/CSE/01

TABLE OF CONTENTS

LIST OF FIGURES

<u>**LIST OF TABLES**</u>

ABSTRACT

Amazon, Google, IBM, Microsoft, and Yahoo are the forerunners that provide cloud computing services. Recently more and more companies such as Sales force, Facebook, YouTube, MySpace etc. also begin to provide different kinds of cloud computing services for Internet users. But at the same time these services also bring some security problems. Currently the majority of cloud computing systems provide digital identity for users to access their services, this will bring some inconvenience for a hybrid cloud that includes multiple private clouds and/or public clouds. Today most cloud computing system use asymmetric and traditional public key cryptography to provide data security and mutual authentication.

Identity-based cryptography has some attraction characteristics that seem to fit well the requirements of cloud computing. In this report, by adopting federated identity management together with hierarchical identity-based cryptography (HIBC), not only the key distribution but also the mutual authentication can be simplified in the cloud.

CHAPTER – 1

INTRODUCTION

GENERAL INTRODUCTION

1.1 CLOUD COMPUTING:

Cloud Computing is a computing model that distributes a large amount of computing resources on a resource pool. With Internet access, users are able to acquire computing resource, storage space and other kinds of software services according to their needs. Cloud computing involves computation, software, data access and storage services that may not require end-user knowledge of the physical location and the configuration of the system that is delivering the services.

The main goal of cloud computing is to make a better use of distributed resources, combine them to achieve higher throughput and be able to solve large scale computation problems. In cloud computing, with a large amount of various computing resources, users can easily solve their problems with the resources provided by a cloud. This brings great flexibility for the users. Using cloud computing service, users can store their critical data in servers and can access their data anywhere they can with the Internet and do not need to worry about system breakdown or disk faults, etc. Also, different users in one system can share their information and work, as well as play games together.

Cloud computing deals with virtualization, scalability, interoperability and quality of services [8]. Cloud computing is best suited for individuals who are seeking a quick solution for startups, such as developers or research projects. Using Cloud computing can help in keeping one's IT budget to a bare minimum. It is also ideally suited for development and testing scenarios. It is the easiest solution to test potential proof of concepts without investing too much capital.

Cloud computing can deliver a vast IT capabilities in real time using many different types of resources such as hardware, software, virtual storage once logged onto a cloud. This allows for cost saving within an organization. Currently Cloud computing clients have to trust 3rd party cloud providers, especially on the availability of cloud service as well as data security.

For users to access the services in a cloud, a user digital identity is needed for the servers of the cloud to manage the access control. While in the whole cloud, there are many different kinds of clouds and each of them has its own identity management system. Thus user who wants to access services from different clouds needs multiple digital identities from different clouds, which will bring inconvenience for users.

1.2 CLOUD COMPUTING SERVICES:

1) Infrastructure as a service (IAAS): Infrastructure as a service (IAAS) provides the required infrastructure as a service. The client need not purchase the required servers, data center or the network resources. The key advantage is that customers need to pay only for the time duration they use the service. As a result customers can achieve a much faster service delivery with less cost. Examples are Go Grid, Flexi scale and Layered Technologies [2].

2) Platform as a service (PAAS): Platform as a service (PAAS) provides a computing platform using the cloud infrastructure. It has all the application typically required by the client. Thus the client need not to buy and install the software and hardware required for it. Through this service developers can get a hold of all the systems and environments required for the life cycle of software, be it developing, testing, hosting of web applications. Key examples are GAE, Microsoft's Azure [2].

3) Software as a service (SAAS): Software as a service (SAAS) eliminates the need to install and run the application on the users system [15]. Important characteristics of this are: [15] Network-based access and management of commercially available software that are managed from centralized locations and enabling customers to access these applications remotely through the internet. Examples of the key providers are SalesForce.com (SFDC), Oracle, IBM and Microsoft [2]. Google Apps is the most widely used SaaS.

1.3 TYPES OF CLOUD:

Currently, there are mainly three types of clouds:

1) **Private clouds:** Private clouds, also called internal clouds, are private networks that offer cloud computing services for a very restrictive set of users within internal network. The main advantage here is that it is easier to manage security, maintenance and upgrades and also provides more control over the use. Private cloud can be compared to intranet. Compared to public cloud where all the resources and applications were managed by the service provider, in private cloud these services are pooled together and made available for the users at the organizational level. The resources and applications are managed by the organization itself. Security is enhanced here as only the organizations' users' have access to the private cloud. For example, some companies and universities can use their internal networks to provide cloud computing services for their own users. These kinds of networks can be thought as private clouds.

2) **Public clouds:** Public clouds or external clouds refer to clouds such as enterprises that provide cloud computing services for the public users. Public cloud allows users' access to the cloud via using web browsers. Users need to pay only for the time duration they use the service, i.e., pay-per-use. This can be compared to the electricity system which we receive at our homes. We pay only for the amount of that we use. The same concept applies here. This helps in reducing the operation costs on IT expenditure. However public clouds are less secure compared to other cloud as all the applications and data on the public cloud are more prone to malicious attacks. The solution to this can be that security checks be implemented through validation on both sides, by the cloud vendor as well as the client. Also both the parties need to identify their responsibilities within their boundaries of operation.

3) **Hybrid clouds:** Hybrid clouds are the clouds that include multiple private and/or public clouds [14]. Providing security in a private cloud and a public cloud is easier, comparing with a hybrid cloud since commonly a private cloud or a public cloud only has one service provider in the cloud. Providing security in a hybrid cloud consisting of multiple service providers is much more difficult especially for key distribution and mutual authentication. It is more secure way to control data and applications and allows the party to access information over the internet. It enables the organization to serve its needs in the private cloud and if some occasional need occurs it asks the public cloud for intensive computing resources.

In public cloud, resources are dynamically provisioned on a self-service basis over the Internet. Services in the cloud are provided by an off-site third-party provider who shares resources and bills on utility computing basis. While in most private clouds, with limited computing resources, it is difficult for a private cloud to provide all services for their users, as some services may require more resources than internal cloud can provide.

Hybrid cloud is a potential solution for this issue since they can get the computing resources from external cloud computing providers. Private clouds have their advantages in corporation governance and offer reliable services, as well as they allow more control than public clouds do. For the security concerns, when a cloud environment is created inside a firewall, it can provide its users with less exposure to Internet security risks.

In private cloud, all the services can be accessed through internal connections rather than public Internet connections, which make it easier to use existing security measures and standards. This can make private clouds more appropriate for services with sensitive data that must be protected.

While in a hybrid cloud, it includes more than one domain, which will increase the difficulty of security provision, especially key management and mutual authentication. The domains in a hybrid cloud can be heterogeneous networks; hence there may be gaps between these networks and between the different services providers. Security can be well guaranteed in each of private/public cloud.

1.4 PROBLEM STATEMENT:

In hybrid cloud with more than one kind of clouds that have different kinds of network conditions and different security policies, how to provide efficient security protection is much more difficult.

For example, cross domain authentication can be a problem in a hybrid cloud with different domains. Although some authentication services such as Kerberos can provide multi-domain authentication, but one of the requirements for the multi-domain Kerberos authentication is that the Kerberos server in each domain needs to share a secret key with servers in other Kerberos domains and every two Kerberos servers need to be registered with each other.

The problem here is if there are N Kerberos domains and each of them wants to trust each other, then the number of key exchanges is N (N-1)/2. For a hybrid cloud with a large number of domains, this will bring a problem for scalability. If different networks in a hybrid cloud using different authentication protocols, this problem can be more complex.

In a cloud, the cloud computing system needs to provide a strong and user-friendly way for users to access all kinds of services in the system. When a user wants to run an application in the cloud, the user is required to provide a digital identity. Normally, this identity is a set of bytes, related to the user. Based on the digital identity, a cloud system can know what right this user has and what the user is allowed to do in the system.

Most of cloud platforms include an identity service since identity information is required for most distributed applications. These cloud computing systems will provide a digital identity for every user.

For example, user with a Windows Live ID can use cloud computing services provided by Microsoft and user who wants to access cloud computing services from Amazon and

Google also needs an Amazon- defined identity and Google account. Here, each of these companies is a public cloud. The problem here is this digital identity can only be used in one private cloud or one public cloud. Users want to access services in the cloud that provided by different clouds will need to have multiple identities, each for one of the cloud. This is obviously not user friendly.

1.5 <u>CHARACTERISTICS OF CLOUD COMPUTING:</u>

1) In cloud computing, users access the data, applications or any other services with the help of a browser regardless of the user's location. The infrastructure which is generally provided by a third-party is accessed with the help of internet.

2) Cost is reduced to a significant level as the infrastructure is provided by a third-party and need not be acquired for occasional computing tasks.

3) Less IT skills are required for implementation.

4) Reliable service can be obtained by the use of multiple sites which is suitable for business continuity [10] and disaster recovery [10].

5) Sharing of resources and costs amongst a large collection of users allows efficient utilization of the infrastructure.

6) Maintenance is easier in case of cloud computing applications as they need not be installed on each user's computer.

7) Pay per use facility allows measuring the usage of application per client on regular bases.

8) Performance can be monitored and thus it is scalable.

9) Security can be as good as or better than traditional systems because providers are able to devote resources to solve security issues that many customers cannot

afford. However, security still remains an important concern when the data is quite confidential.

In a cloud computing system, most data and software that users use reside on the Internet, which bring some new challenges for the system, especially security and privacy. Since each application may use resource from multiple servers. The servers are potentially based at multiple locations and the services provided by the cloud may use different infrastructures across organizations. All these characteristics of cloud computing make it complicated to provide security in cloud computing.

1.6 <u>SECURITY ISSUES:</u>

To ensure adequate security in cloud computing various security issues, such as

1) Identification and Authentication:

In Cloud computing, depending on the type of cloud, specified users must firstly be established and permissions may be granted accordingly. This process is targeting at verifying and validating individual cloud users by employing usernames and passwords protections to their cloud profiles.

2) Authorization:

Authorization is an important information security requirement in Cloud computing. It follows on in exerting control and privileges over process flows within Cloud computing. Authorization is maintained by the system administrator in a Private cloud.

3) Confidentiality:

In Cloud computing, confidentiality plays a major part especially in maintaining control over organization’s data situated across multiple distributed databases. It is a must when employing a Public cloud due to public clouds accessibility nature.

4) Integrity:

The integrity requirement lies within the cloud domain mainly when accessing data. Therefore ACID (atomicity, consistency, isolation and durability) properties of the cloud's data should without a doubt be imposed across all Cloud computing models.

5) Non-repudiation:

Non-repudiation in Cloud computing can be obtained by applying the traditional e-commerce security token provisioning to data transmission within cloud applications such as digital signatures, timestamps and confirmation receipts services (digital receipting of messages confirming data sent/received).

6) Availability:

Availability is one of the most critical information security requirements in Cloud computing because it is a key decision factor when deciding among private, public or hybrid cloud vendors. The service level agreement is the most important document which highlights the availability of resources between the cloud provider and client [9].

All need to be taken into account. Currently, WS-Security service is wildly used in the cloud to provide security for the system. In WS-Security, XML encryption and XML signature are used to provide data confidentiality and integrity. Mutual authentication can be supported by adding X.509 certificate and Kerberos ticket.

1.7 ADVANTAGES OF CLOD COMPUTING:

1) Easy Management

The maintenance of the infrastructure, be it hardware or software is simplified, thus, less headaches for the IT team. Also applications that are quite storage extensive is easier to use in the cloud environment compared to the same when used by the organization by its own. Also at the user level, what you mostly need is a simple web browser with internet connectivity.

2) Cost Reduction

Costly systems need not be required for occasional use of intensive computing resources. Also the man power for such systems is not required. Even simple applications like email can be set up and mostly free through applications like Google Apps.

3) Uninterrupted Services

Cloud computing services provide uninterrupted services to the user. However, some occurrences of outages have occurred in the past, like the Gmail outage in 2009.

4) Disaster Management

In case of disasters, an offsite backup is always helpful. Keeping crucial data backed up using cloud storage services is the need for most of the organizations. Also cloud storage services not only keep your data off site, but they also ensure that they have systems in place for disaster recovery.

5) Green Computing

Harmful emissions due to extensive use of systems in organizations, electronic waste generated as the time passes and energy consumption is the main disadvantage of the present day computing systems. This can be reduced to some extent by using cloud computing services. This leads to environment preserving. Also the e-waste is generated to minimum extent [8].

1.8 <u>PROPOSED SYSTEM:</u>

To solve problems in the cloud, this report proposes to use federated identity management with hierarchical identity- based cryptography (HIBC) in the cloud such that each user and each server will have its own unique identity, and the identity is allocated by the system hierarchically. With this unique digital identity users from a cloud are allowed to access services from other clouds. Using this scheme, the key distribution and mutual authentication in a hybrid cloud can be greatly simplified.

Hierarchy identity-based cryptography is the development from Identity-based cryptography, public key technology that allows the use of a public identifier of a user as the user's public key in order to solve the scalability problem. Identity-based cryptography and hierarchy identity-based cryptography have been proposed to provide security for some Internet applications.

Using federated identity management, each user will have his unique digital identity and with this identity, he can access different services from different clouds.

CHAPTER – 2

LITERATURE SURVEY

2.1 SUMMARY OF PAPERS STUDIED:

PAPER 1

TITLE OF PAPER	Cloud Computing - Concepts, Architecture and Challenges
Author	Kirit Modi & Yashpalsinh Jadeja
Year of Publication	2012
Publication Detail	International Conference on Computing, Electronics and Electrical Technologies [ICCEET]
SUMMARY In this paper we have studied what is the architectural design of cloud computing and its applications. We have also studied a new wave in the field of information technology: cloud computing - its architecture, advantages and some issues. We have approximately infinite computing capabilities, scalability, pay – per - use scheme and so on. However this wave still needs to resolve some of its existing issues with urgency.	

PAPER 2

TITLE OF PAPER	The Management of Security in Cloud Computing
Author	Ramgovind S, Eloff MM & Smith E
Year of Publication	2010
Publication Detail	IEEE

<u>SUMMARY</u>

The purpose of the paper is to provide an overall security perspective of Cloud computing with the aim to highlight the security concerns that should be properly addressed and managed to realize the full potential of Cloud computing. Gartner's list on cloud security issues, as well the findings from the International Data Corporation enterprise panel survey based on cloud threats, is discussed in this paper.

In this paper key security considerations and challenges which are currently faced in the Cloud computing industry are highlighted. While current offerings explore trail-and error control methods, a great deal of investment must be made in the managing security around this evolving technology. The Cloud Security Alliance is one such organization. It is a non-profit organization formed to promote the use of best practices for providing security assurance within Cloud computing, and provide education on the uses of Cloud computing to help secure all other forms of computing.

By following guiding principles discussed in this paper, a great deal of insecurities may be easily expelled, saving business owners' valuable time and investment. Cloud computing has the potential to become a frontrunner in promoting a secure, virtual and economically viable IT solution and future work and progress lies in standardizing Cloud computing security protocols.

PAPER 3

TITLE OF PAPER	Strengthen Cloud Computing Security with Federal Identity Management Using Hierarchical Identity-Based Cryptography
Author	Liang Yan, Chumming Rong, and Gansen Zhao
Year of Publication	2009
Publication Detail	LNCS 5931, pp. 167–177, 2009.

SUMMARY

In this paper, we depicted the principles of identity-based cryptography and hierarchical identity-based cryptography and find the properties of HIBC fit well with the security demands of cloud.

Use of federated identity management and HIBC in the cloud is proposed and depicted how can the system generate and distribute the public and private keys to users and servers. Compared with the current Ws-Security approach, we can see this approach has its advantages in simplifying public key distribution.

Also we studied how the users and servers in the cloud can generate secret session key without message exchange and authenticate each other with a simple way using identity-based cryptography. Also we can see the key escrow problem of identity-based cryptography can be restricted with HIBC approach.

PAPER 4

TITLE OF PAPER	A Survey of Identity-Based Cryptography
Author	Joonsang Baek, Jan Newmarch, Reihaneh Safavi-Naini, and Willy Susilo
Year of Publication	2004
Publication Detail	Proc. of Australian Unix, 2004 - books.google.com

<u>SUMMARY</u>

In this paper, we studied the state of research on identity-based cryptography. Starting from reviewing the basic concepts of identity-based encryption and signature schemes, and subsequently review some important identity-based cryptographic schemes based on the bilinear pairing, a computational primitive widely used to build up various identity-based cryptographic schemes in the current literature.

We also studied the cryptographic schemes such as a certificate based encryption scheme" and a public key encryption scheme, which were able to be constructed.

Finally, we got to know how feasible and under what conditions identity-based cryptography may be used in current and future environments and some interesting open problems concerning with practical and theoretical aspects of identity-based cryptography are proposed.

The state of the art of identity-based cryptography is discussed throughout the paper; there are pros and cons of using identity-based cryptography. From the authors' point of view, identier information that will be used as public key in identity-based cryptography and management of them are important next steps that cryptographers and security engineers should elaborate on.

PAPER 5

TITLE OF PAPER	Identity-Based Encryption from the Weil Pairing
Author	Dan Boneh & Matthew Franklin
Year of Publication	2003
Publication Detail	Appears in SIAM J. of Computing, Vol. 32, No. 3, pp. 586-615, 2003.

<u>SUMMARY</u>

We studied chosen cipher text security for identity-based systems and a fully functional identity-based encryption scheme (IBE) system is proposed. The system has chosen cipher text security in the random oracle model assuming BDH, a natural analogue of the computational Diffie-Hellman problem. The BDH assumption deserves further study considering the powerful cryptosystems derived from it. For example, it could be interesting to see whether the techniques can be used to prove that the BDH assumption is equivalent to the discrete log assumption on the curve for certain primes p.

Cocks recently proposed another IBE system whose security is based on the difficulty of distinguishing quadratic residues from non-residues in the ring Z=NZ where N is an RSA modulus (i.e., a product of two large primes). Cocks' system is somewhat harder to use in practice that the IBE system in this paper.

Cocks' system uses bit-by-bit encryption and consequently outputs long cipher texts. Also, encryption/decryption is a bit slower than the system described in this paper.

Nevertheless, it is encouraging to see that IBE systems can be built using very different complexity assumptions. It is an open problem to build chosen cipher text secure identity based systems that are secure in the standard computation model (rather than the random oracle model). One might hope to use the techniques of Cramer-Shoup to provide chosen cipher text security based on DDH.

Unfortunately the DDH assumption is false in the group of points on the curve E. However, simple variants of DDH do seem to hold. Building a chosen cipher text secure IBE in the standard model is currently an open problem. Precise definitions for secure identity based encryption schemes and several applications for such systems are also defined.

2.2 INTEGRATED RESEARCH SURVEY:

Hash functions

$$x = H(m)$$

Hash functions are functions which map *(hash)* a message to a collision free value (also called a hash value) and so this new, usually shorter, value can be used as a representation of the original message. Hash functions without full uniqueness are commonly used in other fields of computer science but for cryptography it is 'must have' property that the hash value is unique.

A useful hash will hash arbitrary sized messages to a usually much-condensed fixed sized hash, typically 160 bits. Commonly used hash functions are the relatively fast MD5 and SHA-1. Recently MD5 has fallen out of favor because of recently exposed weaknesses in its ability to always product a strong hash and is only now used in legacy systems.

One way hashes are hash functions where given the hash of a message and the algorithm, it is impossible to find the original message.

Prime numbers

A prime number is a number that has exactly two positive integer factors, 1 and itself.
The problem is no one has yet figured out a computationally efficient method to say if a number is neither a prime nor how to predict what the next prime number will be. The only known method is brute force (exhaustive search) i.e. going through all possibilities.

Groups

A group is a set of numbers with an operator. A relatively simple group to illustrate the properties useful for cryptography is numbers from 1 to 4 and an operator multiply x *mod* 5.

[X 1, 2, 3, 4 *mod* 5]

1) The group is closed when the answer is always within the group (3 x 2) = 6 *mod* 5 = 1

2) The group has an identity which is a number which when applied by the operator does not change the element. 2 x 1 = 2

3) It is associative (2 x 3) x 4 = 2 x (3 x 4) *mod* 5

4) Every element has an inverse
   (3 x 2) = 6 = 1 *mod* 5
   2 is the inverse of 3

5) A group is cyclic if it has a member that when subjected to repeated applications of the operator will give every member of the set
   2 = 2 *mod* 5 = 2
   2 x 2 = 4 *mod* 5 = 4
   2 x 2 x 2 = 8 *mod* 5 = 3
   2 x 2 x 2 x 2 = 16 *mod* 5 = 1

This is easy since this group has 4 members.
The point here is that applying the operation *N* times is easy. Finding out how many times an operation was applied is computationally too expensive. This is a one way function.
$\boldsymbol{y = g^x \textbf{ mod } p}$

The formula can be used to generate elements of the group over the field F_g. In this context *g* can be described as the generator of the group. The number of elements in the group will be a divisor of *p* - 1. In our example 2 is a generator of group of order 4 over F_5.

Random Numbers

Randomness is an important concept inside a security. Strong random numbers are crucial to the security of the mathematics used in cryptography. Random numbers are often used as keying material to encryption algorithms. So non-randomness can result in predictable keys and lead to security weaknesses. To overcome this limitation various techniques are used to substitute for the deterministic issues.

A critical function of any crypto related applications is the secure treatment of the keys used in the cryptographic operations. While most emphasis is on the secure storage or transport of keys. There also exists a requirement to stop an attacker gaining access to the keys by either deducing or by guessing what they could be. Even a guess which reduces the time or computation required for a brute force attack is a risk. If the generation of keys is not truly random then even the most advanced key protection methods will fail. One has to rely on pseudo random events that hold statistical properties which overwhelm any attempt to pre-empt or guess their outcome.

Most programming languages do provide a random function which can be suitable for applications such as games. However for cryptographic purposes this function is weak. For example, consider functions based on some initial value called the seed. When given the same seed, two different instances of a program utilizing rand () will produce the same random values.

In many implementations of C, if a seed is not explicitly specified, it is calculated from the current value of the system timer which is not considered genuinely random input as it is very easy to guess if you know when a protocol took place. To overcome this we can mix numerous sources of data into one virtual data source taking care that in doing so we don’t weaken the outcome. The most common ‘mixer’ is use of a good, fast hash function such as MD5 or SHA1.

Linux and most UNIX operating systems use the steps outlined to take care of the generation of cryptographic secure entropy at the kernel layer. Linux for example can be configured so that at boot time the random pseudo device driver will only be active (and allow use) if the underlying device has gathered statistically enough random data to allow for its secure use. The UNIX implementation uses this driver as a random source.

2.3 <u>Types of cryptography</u>

There are two main types of mathematical cryptography

1) Symmetric or *secret* key
2) Asymmetric or *public* key

2.3.1 Symmetric Key

Symmetric key is an easier to understand concept. Essentially one party (the encryptor) uses a secret key to a mathematical function which encrypts the plaintext message to a secure form. The message is passed to the decrypter who uses ***the same key*** to an inverse mathematical function which decrypts the message returning it to its original plaintext.

The principal issue in symmetric cryptography is the secure transport of the key between the parties. Examples include DES and its variants, IDEA, & AES.

2.3.2 Asymmetric Key

Asymmetric key is generally broken up into two parts known as the *public* key and the *private* key. The public is used to encrypt the message and the private to decrypt. Asymmetric is popularly known as *public key cryptography.*

The advantage of asymmetric cryptography lies with the property that possessing the public key provides no clue to the private key allowing the public key to be freely published to allow anyone to encrypt to the holder of the private key. This unique property overcomes the limitations of symmetric cryptography which requires prior 'key swapping before use. Asymmetric algorithms are much more computationally expensive than symmetric and are not suitable to encrypt large amounts of data.
The solution to practical use is using a combination of both methods to provide both practical performances together with the benefits of usability.

2.4 <u>Commonly Used Symmetric Algorithms</u>

2.4.1 DES

The Data Encryption Standard (DES) is currently being replace by its successor AES. It has emerged that the team behind the creation of DES have admitted that defending against differential cryptanalysis was a design goal of DES. DES has been extensively studied since it was so widely adopted. It is easily the best known and most widely used symmetric algorithm.

Technical details

DES used a 64—bit block size (breaks data into 64— bit chunks before encryption) and uses a 56—bit key during execution (8 parity bits are removed from the 64—bit key). Data passes through 16 rounds of byte substitution (S-boxes). Like most symmetric block ciphers DES can operate in a number of modes of operation, the simplest being electronic codebook ECB where each block is encrypted with the same key. This has weaknesses where identical message blocks encrypted with the same key result in the same cipher text output.

Using DES in the wrong mode opens a weakness which would allow an attacker to compromise the communications by 'injecting' malicious data into the data stream.

An improvement to the original design is to use a 128—bit key (actually 112—bit because of the parity stripping) and, using a first 56-bit to encrypt, a second 56-bit key to decrypt and finally the first key again to encrypt each block twice (as secure as 3 keys) the resulting 'triple DES' was the preferred mode of operation until AES was accepted.

2.4.2 AES

Triple DES governing rule

- Symmetric-key
- Minimum block sizes of 128,
- Minimum keys sizes of 128—, 192— and 256—bits
- More efficient than triple DES

- Publicly disclosed
- Available freely worldwide.

We use AES exclusively as the preferred symmetric algorithm in our IBE subsystem.

Technical Details

AES follows the minimum specification detailed, any combination of allowed key size and fixed block size of 128-bits are usable. It is a block cipher unlike DES with rounds of operations on blocks. Operations used in AES are byte-shuffling operations. Addition corresponds to the bitwise XOR operation. The multiplication operation in this is harder, and often implemented with simple lookup tables. AES does no arithmetic operations. These properties allow for very efficient implementation in hardware.

A simplified description: The basic operation is that of data processed by transformations via 'round' operations on data stored in a four a four matrix. Each round 1 is a set of 4 operations on the matrix. The number of rounds depends on the key size chosen. The 4 operations in each round are Byte Substitution (like a DES S-box), Mix-Column, Shift-Row and Round-Key Addition.

2.5 <u>Commonly Used Asymmetric Algorithms</u>

2.5.1 RSA

Discovered in 1977 and named after its inventors Ron Rivest, Adi Shamir and Leonard Adleman. RSA encryption is based on factoring and transforms the message"M" into"C" with the formula

$$C = M\ e \bmod N$$

The numbers e and *N* are the two public numbers we create and publish. They are ''public key." The message M can be simply the digital value of a block of ASCII characters.

The formula says: multiply the Message *M by* itself e times, then divide the result by the number *N* and save only the remainder. The remainder that we have called C is the encrypted representation of the message.

Technical Overview

Key Set-up

1. Choose two large random prime numbers p and q such that $|p| < |q|$.
2. Compute $N = p \times q$
3. Compute $\phi(N) = (p - 1) \times (q - 1)$
4. Choose a random integer $e < \phi(N)$ such that gcd $(e, \phi(N) = 1)$ and compute an integer d such that $ed = 1 \ (mod\ (\phi\ (N)\ ))$
5. Publicize (N, e) as public key, discarding p, q and $\phi\ (N)$ and keep d as private key. e can be small but d must be impossible to guess.

Send a message $M < N$ to Alice, the sender Bob creates a cipher text C by

$$\boldsymbol{C \leftarrow M\ e\ (\textbf{mod}\ N)}$$

Alice computes

$$\boldsymbol{M \leftarrow C\ d\ (\textbf{mod}\ N)}$$

2.5.2 ECC

Another form of ‘hard problem’ that mathematicians have found useful to the field of cryptography is that of Elliptic Curves.
The discovery of the use of Elliptic Curves for public key cryptography can be attributed independently to Neil Koblitz and Victor Miller who both made discoveries in 1985.

Technical Overview

Curves can be defined in Affine (2 dimensions) or Projective (3 dimensions coordinates). An elliptic curve is a graph (curve) which can be defined by equations of the form.

$$\boldsymbol{y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6}$$

All a’s are constant. Formally this is described as an *elliptic curve E* (F_q), and is the set of solutions (x, y) over curve to an equation of the form

$$\boldsymbol{E: y^2 + a_1xy + a_2y = x^3 + a_2x^2 + a_4x + a_6}, \qquad (1)$$

Where a_i belongs to F_q and there also exists a *point at infinity* denoted *O*.

The special properties are that elliptic curves behave ‘well’ when operations are performed with a prime modulus greater than 3 and any elliptic curve of the form (1) can be converted by the transformation to

$$y^2 = x^3 + a_4x + a_6$$

To follow general cryptographic convention a_4 is renamed a and a_6 is renamed b.
This gives us the curve

$$y^2 = x^3 + ax + b \bmod p$$

This means we are only allowed to use the integers from zero to p - 1 as input.

For example let us take an equation with p = 11, a = 4 and $b = 7$.

$$y^2 = x^3 + 4x + 7 \bmod 11$$

Plug in all values for x and we get the associated values for y plus the point at infinity (denoted by O). Now any pair of points that satisfy the equation can be used.

To add two points on a curve we can’t simply add the coordinates to find a point which still satisfies the curve.
However there are a set of rules which one can apply for curves of this type.
The rules are

- **Rule 1:** $O + O = O$
- **Rule 2:** $(x_1, y_1) + O = (x_1, y_1)$
- **Rule 3:** $(x_1, y_1) + (x_1, -y_1) = O$

Using these rules two points on the curve may be added to yield a third point also on the curve.

***Note 1*:** If we ever, in our calculations, divide by zero we can stop and say the result is the point at infinity (O).

Note 2: The point at infinity (O) acts like zero in regular addition: Add a point to the point at infinity we get the original point. This is the additive identity for the group.

***Note 3:* Prime Modulus**

The group is cyclic, it has a generator function such that when this function is applied to any member of the group, and it will only result in another member of the group. If applied to all members of the group then it will produce another set containing all members of the original group. Interestingly the output sequence of the results of the application of the generator holds a random distribution.

***Note 4:* Multiplication**

Take a point *P,* an *(x, y)* point and multiply it by an integer *d.* we can break down $d* P$ to $(P + P + ...P)$ i.e. *P* added to itself *d* times. Then apply the rules of addition. This integer *d* is called a *scalar* as opposed to the coordinate (point) *P*.

2.6 What does all this mean to cryptography?

We'll take an elliptic curve (i.e. modulus *p* and parameters *a* and *b)* and a point on this curve *P*. Then take a scalar *d* and find $d* P$ to get another point on the curve *Q.*

We keep *d* secret. We can use the curve *(p, a, b)* and points *P, Q* as the public key. The challenge for any attacker is to find *d.* No-one has found a sub-exponential algorithm to be able to compute *d.*

If the modulus *p* is large enough (200 bits or so. a big number ....) then it would take today's supercomputers many thousands of years. This is the basis for the application to cryptography. Scalar multiplication on an elliptic curve is relatively easy, but the inverse, which is a discrete logarithm problem, is extremely hard. To illustrate we will consider Key agreement using Elliptic Curve Diffie-Hellman.

Diffie-Hellman is a technique to allow unauthenticated key agreement using exponentiation. Its security rests on the intractability of the Computational Diffie-Hellman problem and the Discrete Logarithm Problem.

- Alice calculates curve and makes *p, a, b* and a point *P* public.
- She generates some random d_a and keeps this secret.
- Alice sends Bob Q_a which is equal to d_aP.
- Bob gets Alice's public components and generates his own random d_b.

- He calculates Q_b by computing d_bP.
- Bob then computes a secret value $S = d_bQ_a$. Since Q_a is just d_aP what Bob has computed is $S = d_bd_aP$.

He sends Alice Q_b and Alice uses this to compute her secret value $S = d_aQ_b$ since Q_b, is d_bP, what Alice has done is to compute $S = d_ad_bP$.
This is the same as Bob computed!
So Alice and Bob can 'secretly' get to the same point on the curve S, by using this.

A simple method to extract a key is just to ignore the y coordinate and take the x coordinate as a number. This derived number can be used as a key. Bob can use this secret value to make an AES encryption key. Alice can use the method outlined above to get same encryption key. So what Bob encrypts, Alice can decrypt. The Attacker Eve intercepting this exchange just knows p, a, b, P, Q_a, and Q_b. The only way for Eve to figure out S is to get either d_a or d_b which Alice and Bob are holding secret.
Apparently the only way she can get a d is to calculate one of the $d's$ is by using the fact that she knows either $Q_a = d_aP$ and she knows Q_a and P

OR

$Q_b = d_bP$ and she knows Q_b and P.

This is exactly the discrete logarithm problem of ECC and this is computationally unfeasible with today's technology and for the foreseeable future.

2.7 Pairings

Joux and Sakai, Ohgishi & Kasahara [8] independently proposed using properties of pairing-mapping functions applied to cryptography.

Let G_1 and G_2 denote two groups of prime order q, where G_1, with an additive notation, denotes the group of points on an elliptic curve; and G_2, with a multiplicative notation, denotes a subgroup of the multiplicative group of a finite field.

Multiplicative groups will be represented here as Z_n^*, which is the set of positive integers less than *n* and relatively prime to *n,* under multiplication modulo *n.* An integer is relatively prime to another if their only common positive divisor is 1. For example, 8 and 15, even though they are not prime numbers, are relatively prime.

A pairing is a computable bilinear map between these two groups. Two pairings have been studied for cryptographic use. They are the Weil pairing and the Tate pairing. ê denote a general bilinear map, i.e. ê: G_1 x $G_1 \rightarrow G_2$, which can be either a modified Weil pairing or a Tate pairing.

2.8 <u>Computation Expense</u>

To perform either RSA or elliptic curve cryptography on a large set of data is very computationally expensive and would take too long a time for real-time data communications such as an email system.

The solution is to use some other more efficient algorithm like a symmetric algorithm such as AES or DES and perform the bulk data encryption and use the public key methods (ECC or RSA) to encrypt the key(s) used by this efficient algorithm and transport them with (usually by simply attaching them to) the encrypted data.

CHAPTER – 3

ANALYSIS, DESIGN AND MODELING

3.1 IDENTITY-BASED CRYPTOGRAPHY AND SIGNATURE

Identity-based cryptography and signature schemes were firstly proposed by Shamir [11] in 1984.

Identity-based cryptographic scheme is a kind of public-key based approach that can be used for two parties to exchange messages and effectively verify each other's signatures. Unlike in traditional public-key systems that using a random string as the public key, with identity-based cryptography users identity that can uniquely identify that user is used as the public key for encryption and signature verification. Identity based cryptography can ease the key management complexity as public keys are not required to be distributed securely to others. Another advantage of identity-based encryption is that encryption and decryption can be conducted offline without the key generation center.

In this system our friends Alice and Bob wants to communicate. Bob can simply use Alice's email address (alice@wonderland.com) and the public parameters of a trusted third party as the public key to encrypt the message. When Alice receives the message, she contacts the trusted third party (PKG private key generator), validates herself and receives the private key associated with her identity.

Two Japanese researchers Sakai and Kasahara [14] also made a significant discovery relevant to IBE using pairings but due to language issues their implementation wasn't widely known. But only in 2001, an efficient approach of identity-based encryption schemes was developed by Dan Boneh and Matthew K. Franklin [13] and Clifford Cocks [12] using the Weil pairing. These schemes are based on bilinear pairings on elliptic curves and have provable security.

In the identity-based cryptography approach, the PKG should creates a "master" public key and a corresponding "master" private key firstly, then it will make this "master" public key public for all the interested users. Any user can use this “master” public key and the identity of a user to create the public key of this user. Each user wants to get his private key needs to contact the PKG with his identity. PKG will use the identity and the "master" private key to generate the private key for this user.

In Dan Boneh and Matthew K. Franklin’s approach, they defined four algorithms for a complete identity-based cryptography system.
It includes

(i) **Setup,**
(ii) **Extract,**
(iii) **Encryption and**
(iv) **Decryption.**

3.2 IBE ADVANTAGES

The main advantage to IBE is

1) Ability of one end entity to generate a public key of another’s without having to find it on the CA’s infrastructure or from the other entity in a prior transaction. This saves on bandwidth requirements on the client and saves the cost of always-on, highly-available, securely hosted servers available in a many-to-one model. In comparison all the IBE PKG needs to do is distribution of private keys in a one-to-one setting.

2) In IBE the server can choose only to work with paying registered clients and has no reduction on the security of the model.

3) Another advantage is the fact that the keys are generated centrally, and that keys can be recreated on demand by re-entering the original parameters. When there are legal demands for retaining keys, like those imposed by US regulatory bodies on public companies, IBE makes it extremely simple to manage the potential large key set an organization generates over time. Only one secret needs to be secured.

3.3 IBE DISADVANTAGES

1) One disadvantage of IBE is the requirement that the channel over which an identity's private key is transported from the PKG to the identity needs to be secure.

2) Another potential problem of the identity-based cryptography is the **revocation problem.** Because all the users in the system use some unique identifiers as their public keys. If, for example, Bob loses his private key, does Bob need to change identity? Or if the public key is the user's name, address, or email address, it is inconvenient for the user to change it. The solution to revocation problem is to modify the ID public key string to add time window based keys. But the management of the increased number of keys adds some overheads to the end user.

3) One of the issues is the **key escrow problem**. Since users' private keys are generated by PKG, the PKG can decrypt a user's message and create any user's digital signature without authorization. This in fact means that PKGs must be highly trusted.

So the identity based scheme is more appropriate for a closed group of users such as a big company or a university. Since only under this situation, PKGs can be set up with users' trust.

For a system using identity-based cryptography, key escrow problem is inherent and cannot be avoided since PKG knows the private keys of all the users. But this problem can be overcome by methods like splitting up the master secret or CA keys among a number of parties. An alternative is a ***hierarchical*** **IBE** where the actual keys are created not by the root PKG but by sub PKG's can improve the scalability of traditional identity-based cryptography scheme.

3.4 HIERARCHICAL IDENTITY BASED CRYPTOGRAPHY

In hierarchical identity-based cryptography system, only the PKG in the same domain as the users can knows their private keys. PKGs in other domains or at other levels cannot know these private keys, so that the key escrow problem can be restricted in a small range.
In a HIBC network, a root PKG will generate and distribute private keys for domain-level PKGs and the domain-level PKGs will generate and distribute private keys to the users in their own domain. HIBC is suitable for a large scale network since it can reduce the workload of root PKG by distribute the work of user authentication, private key generation and distribution to the different level of PKG's. It can also improve the security of the network because user authentication and private key distribution can be done locally.

The HIBC encryption and signature algorithms include

- Root setup,
- Lower-level setup,
- Extraction,
- Encryption and
- Decryption.

In a system using HIBC, every PKG in the hierarchy knows the user's private keys in the domain under the PKG. Although key escrow problem cannot be avoided, this can limit the scope of key escrow problem. PKG has the ability to recreate the keys if it so wishes and it therefore decrypt any IBE secured data. This ability of the PKG to decrypt all data makes it a vulnerable point of attack and subsequently it has to be managed in a highly secure manner. Using HIBC in the cloud, an important part is key generation and distribution [1]. The security of HIBC scheme is based on the using of admissible pairing.

3.4.1 Root setup algorithm:

- **(i)** Root PKG will generate the root PKG system parameters and a root secret.
- **(ii)** The root secret will be used for private key generation for the lower-level PKG's.
- **(iii)** The root system parameters are made publicly available and will be used to generate public keys for lower-level PKGs and users.

3.4.2 Lower-level setup algorithm:

- **(i)** Each lower-level PKG will get the root system parameters and generate its own lower-level secret.
- **(ii)** This lower-level secret will be used to generate private keys for the users in its domain.

3.4.3 Extract Algorithm:

When a user or PKG at level t with its identity ($ID_1 \ldots\ldots ID_t$) requests his private key from its upper-level PKG, where ($ID_1 \ldots. ID_i$) is the identity of its ancestor at level i (1 <= i <= t), the upper-level PKG will use this identity, system parameters and its own private key to generate a private key for this user.

3.4.4 Encryption Algorithm:

User who wants to encrypt a message M can use the system parameters, receiver's identity and the message as input to generate the cipher text.

C = Encryption (parameters, receiver ID, M).

3.4.5 Decryption Algorithm:

Receiving a cipher text, receiver can use system parameters and his private key got from the PKG to decrypt the cipher text.

M = Decryption (parameters, k, C), k is the private key of the receiver

3.4.6 Signing Algorithm:

A user can use parameters, its private key, and message M to generate a digital signature and sends to the receiver.

Signature = Signing (parameters, k, M), k is the sender's private key.

3.4.7 Verification Algorithm:

Receiver verify the signature using the parameters, message M, and the sender's ID.

Verification = (parameters, sender ID, M, Signature).

3.5 Federated Identity Management in Cloud

Compared with centralized identity, which is used to deal with security problems within the same networks, federated identity is adopted to deal with the security problem that a user may want to access external networks or an external user may want to access internal networks.

Federated identity is a standard-based mechanism for different organization to share identity between them and it can enable the portability of identity information to across different networks. One common use of federated identity is secure Internet single sign-on, where a user who logs in successfully at one organization can access all partner networks without having to log in again.

Using identity federation can increase the security of network since it only requires a user to identify and authenticate him to the system for one time and this identity information can be used in different networks. Use of identity federation standards cannot only help the user to across multiple networks include external networks with only one time log in, but also can help users from different networks to trust each other.

Using identity federation in the cloud means users from different clouds can use a federated identification to identify themselves, which naturally suit the requirement of identity based cryptography in cloud computing.

In our approach, users and servers in the cloud have their own unique identities. These identities are hierarchical identities. To access services in the cloud, users are required to authenticate themselves for each service in their own clouds. In some cases, servers are also required to authenticate themselves to users. In a small and closed cloud, this requirement can be satisfied easily.

While in a hybrid cloud, there are multiple private and/or public clouds and these clouds may rely on different authentication mechanisms. Providing effective authentications for users and servers from different cloud domains would be difficult. In the cloud trusted authority PKGs are used and these PKGs will not only act as PKGs in traditional identity-based cryptography system but also allocate hierarchical identities to users in their domains.

There is a root PKG in overall domain of each cloud, and each sub-level domain (private or public cloud) within the cloud also has its own PKG. The root PKG will manage the whole cloud, each private cloud or public cloud is the first level and users and servers in these clouds are the second level. The root PKG of the cloud will allocate and authenticate identities for all the private and public clouds. Each private cloud and public cloud uses its own domain PKG to allocate and manage the identities of all the users and servers in its own cloud. Each user and server in this domain has its own identity and this identity is a hierarchical identity, which includes both the identity of the user or server and the identity of the domain.

3.6 Data Encryption and Digital Signature

In the cloud, one of the most important security problems are mutual authentication between users and servers, protection of data confidentiality and integrity during data transmission by encryption using secret keys.

Using federated identity, any user and server has its unique identity and any user and server can get the identity of any other user/server by request with the PKG’s.

With HIBC, the public key distribution can be greatly simplified in the cloud. Users and servers do not need to ask a public key directory to get the public key of other users and servers as in traditional public key schemes. If any user or server wants to encrypt the data that transmitted in the cloud, the sender can acquire the identity of the receiver, and then the sender can encrypt the data with receiver's identity.

Currently, WS-Security (Web service Security) protocol which can provide end-to end message level security is widely applied in cloud computing to protect the security of most cloud computing related web services. Ordinarily XML message representation is about 4 to 10 times large compared with their equivalent binary formats.

3.7 Secret Session Key Exchange and Mutual Authentication

Identity-based cryptography is a public key cryptography scheme; it is much slower when it is compared with symmetric key cryptography. In practice, public key cryptography is not used for data encryption in most of the clouds. For example, in XML encryption, XML data is encrypted using symmetric cryptography such as AES and Triple-DES. This secret symmetric key is encrypted using the public key encryption and then transmitted to the receiver.

While in the cloud with HIBC, this secret symmetric key distribution can be avoided since identity-based cryptography can be used for secret session key exchange. For every two parties in the system using identity-based cryptography, it is easy for each one of the two parties to calculate a secret session key between them using its own private key and public key of other party; this is called identity-based non-interactive key distribution.

In a cloud using HIBC, each user or server can calculate a secret session key between it and the other party it wants to communicate with without message exchange. This advantage of identity-based cryptography can not only reduce message transmission but also can avoid session key disclosure during transmission. This secret session key can be used not only for data encryption, but also for mutual authentication [7].

We assume if a user with identity Alice@UiS and a server with identity Storage@google in the cloud want to authenticate each other. First, they can calculate a secret session key K_s between them. Then Alice can send a message to the server as:

Alice →Server: Alice @ UiS, M, f (Ks, Alice @ UiS, Storage @ google, M)
M is a randomly selected message and f is a one way hash function. Here, to compute the correct hash value, a correct secret session key K_s is needed. Since K_s computation requires Alice's private key and this private key can only be allocated from the PKG in the private cloud of University of Stavanger, thus Alice can be verified that she is a legal user of this cloud. Also the server can authenticate itself to Alice the same way. We can notice that this mutual authentication does not include any certification from a third party.

3.8 <u>FUNCTIONAL REQUIREMENTS:</u>

The desirable design requirements are

- The senses of human beings are not accurate enough to distinguish minor changes in message.
- It provides encryption & decryption of message.
- It does not compromise on privacy and security.
- Code be portable

3.9 <u>NON – FUNCTIONAL REQUIREMENTS:</u>

1) Performance Requirements

A Performance requirement is the extent to which a function must be executed, and is generally measured in terms of quantity, quality, coverage, timeliness or readiness. Performance requirements are initially defined through requirement analyses and trade studies using customer need, objective, and/or requirement statements.

2) Safety Requirements

Technical feasibility such as whether the necessary technology exist to do what is suggested and if the equipment have the capacity to hold data required by the use of new system. Also any technical guarantees of accuracy, reliability, ease of access, data security could be provided etc. This is the perceived risk of possible damage to property, people and the environment. It is important to highlight and understand the potential damage that could occur when using a product within the anticipated operational environment.

3) **Security Requirements**

Security is an emergent quality of software, much like usability and performance, in that it comes together as the team implements all the functionalities. Requirements for emergent qualities typically strive to impose some minimum standards on the software or specifically prohibit unwanted behavior, rather than stating expected software behavior like for functional requirements.

<u>Block diagram for identity based cryptography:</u>

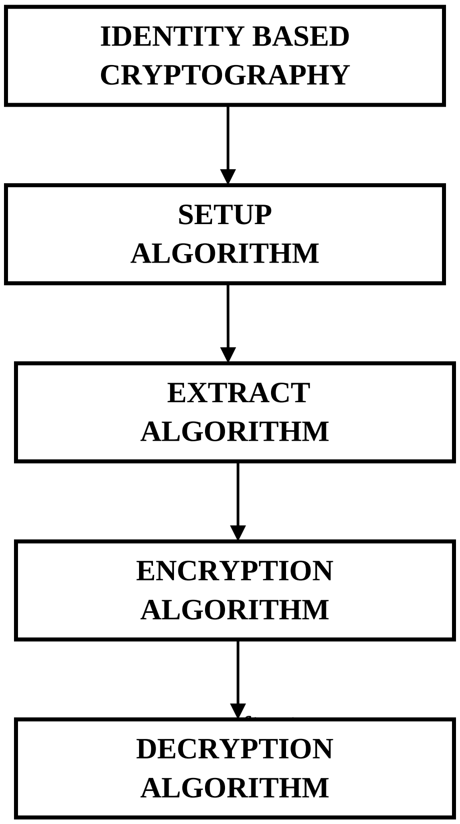


Figure 1: Identity based cryptography

Flowchart for Setup algorithm:

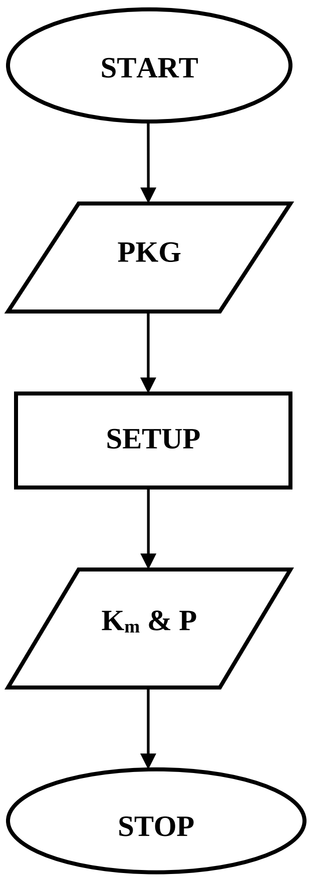


Figure 2: Setup algorithm

Flowchart for Extract algorithm:

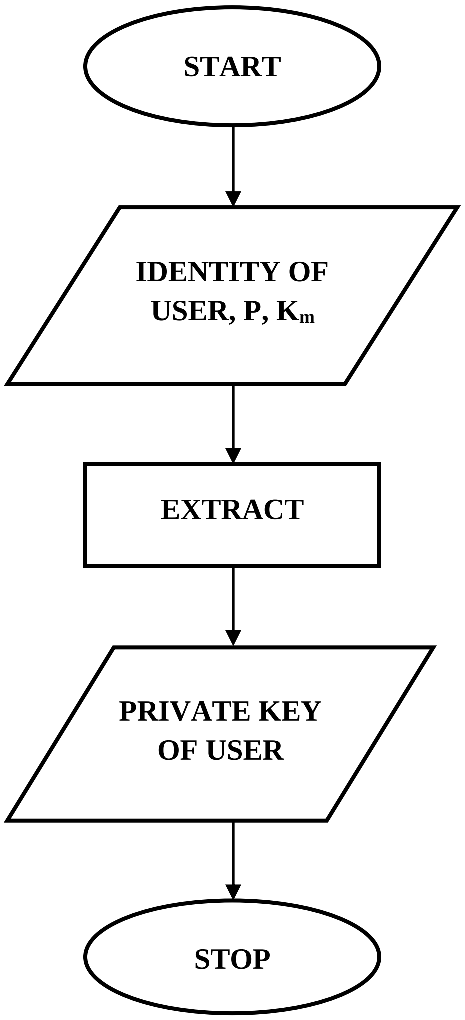


Figure 3: Extract algorithm

<u>Flowchart for Encryption algorithm</u>:

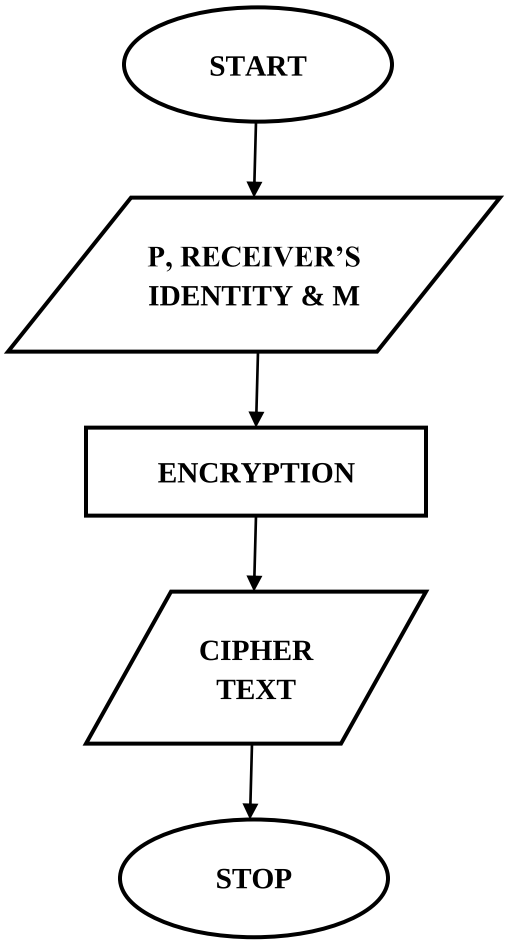


Figure 4 Encryption algorithm

<u>Flowchart for Decryption algorithm</u>:

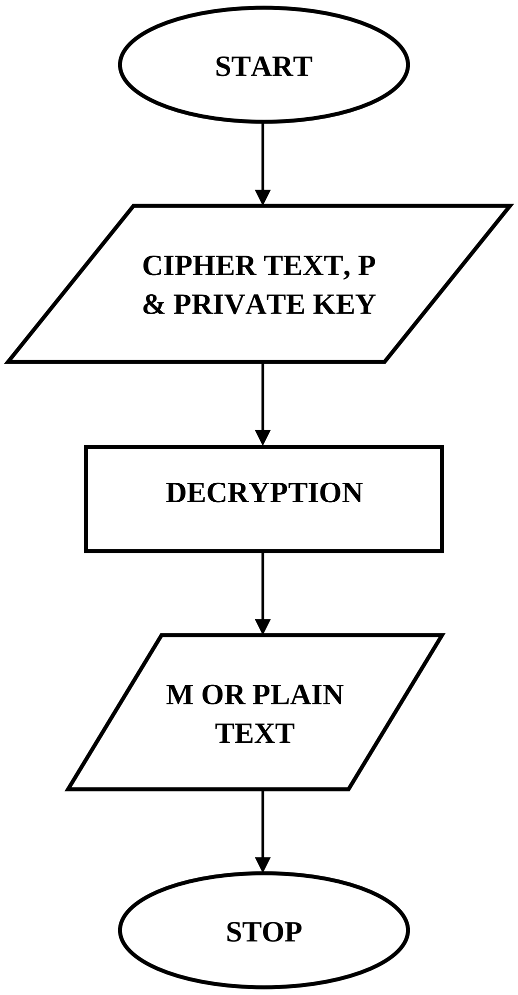


Figure 5: Decryption algorithm

Block diagram for HIBC Cryptography & Signature:

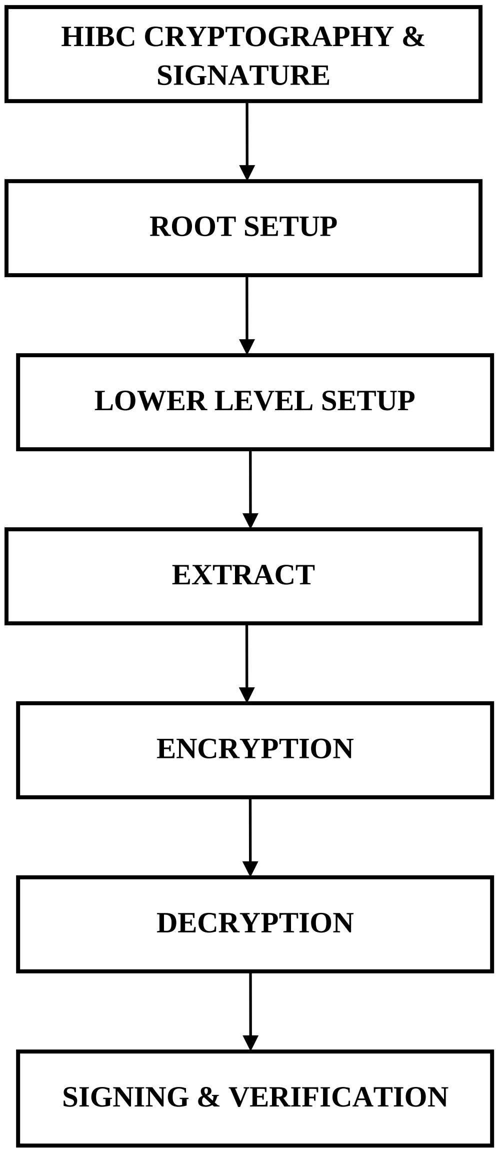


Figure 6: HIBC Cryptography & Signature

<u>Flowchart for Root Setup Algorithm:</u>

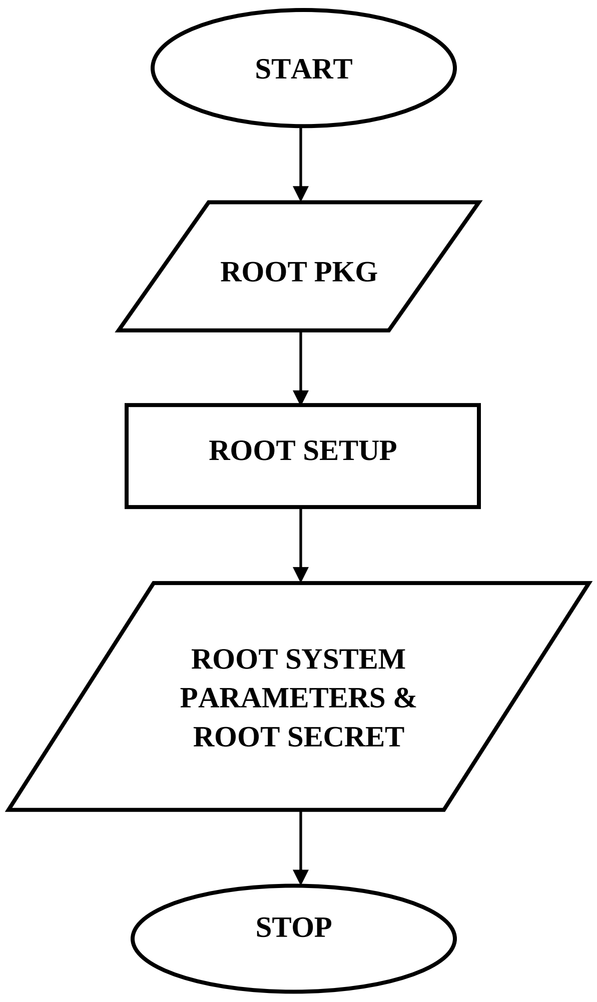


Figure 7: Root Setup Algorithm

<u>Flowchart for Lower - Level Setup Algorithm:</u>

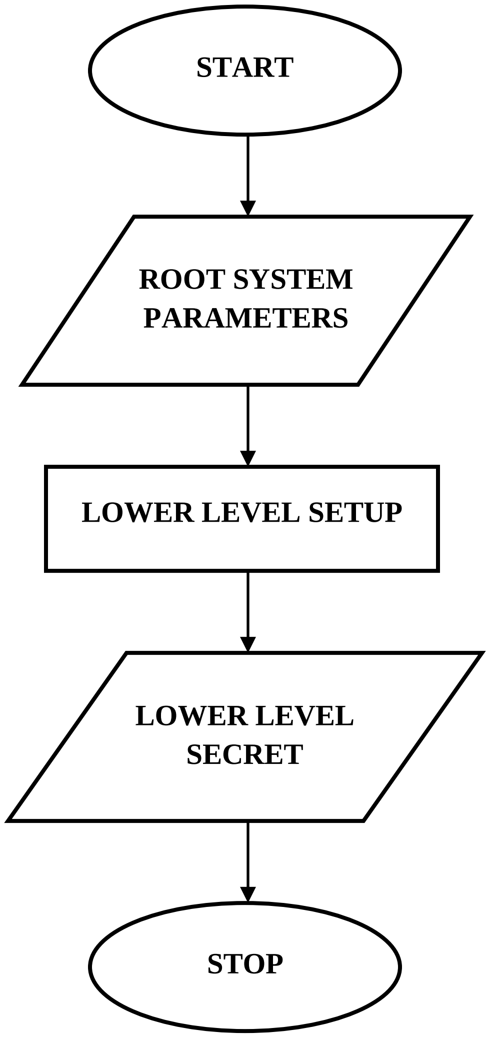


Figure 8: Lower - Level Setup Algorithm

Flowchart for Extract Algorithm:

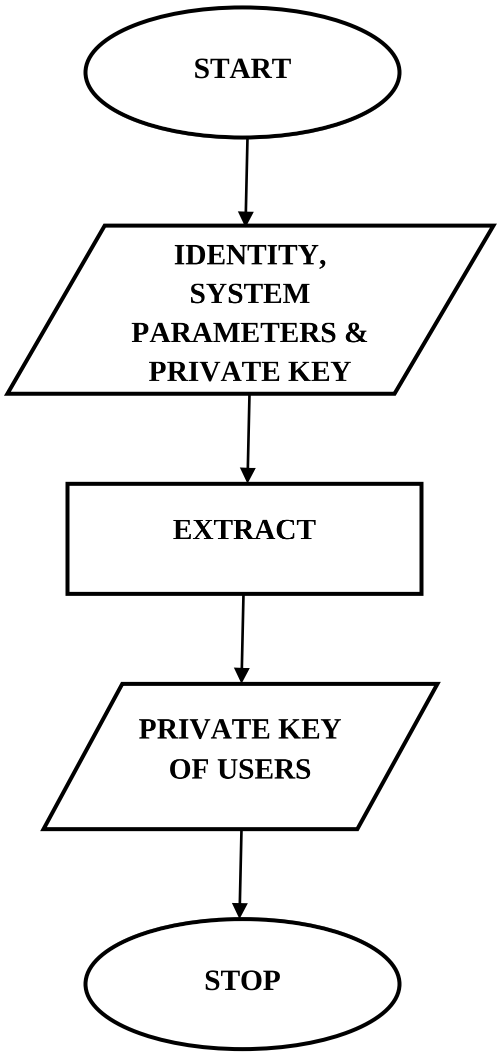


Figure 9: Extract Algorithm

<u>Flowchart for Encryption Algorithm:</u>

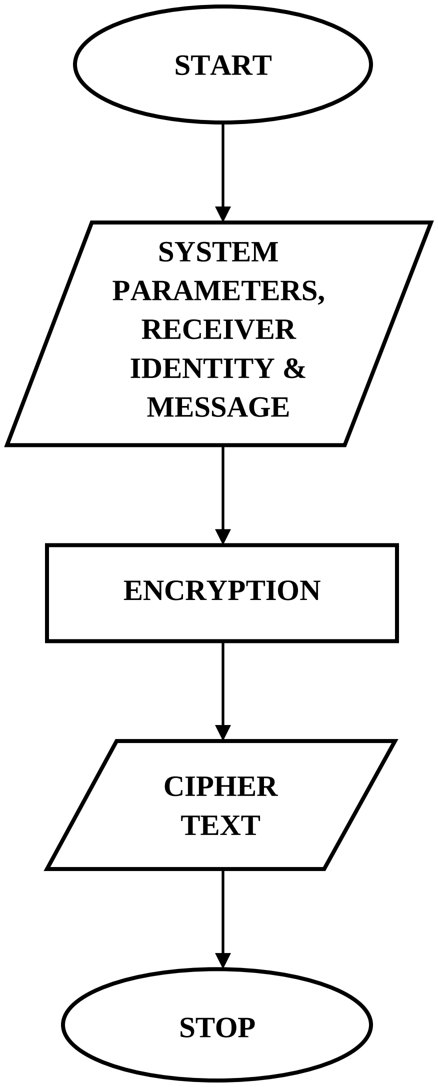


Figure 10: Encryption Algorithm

<u>Flowchart for Decryption Algorithm</u>:

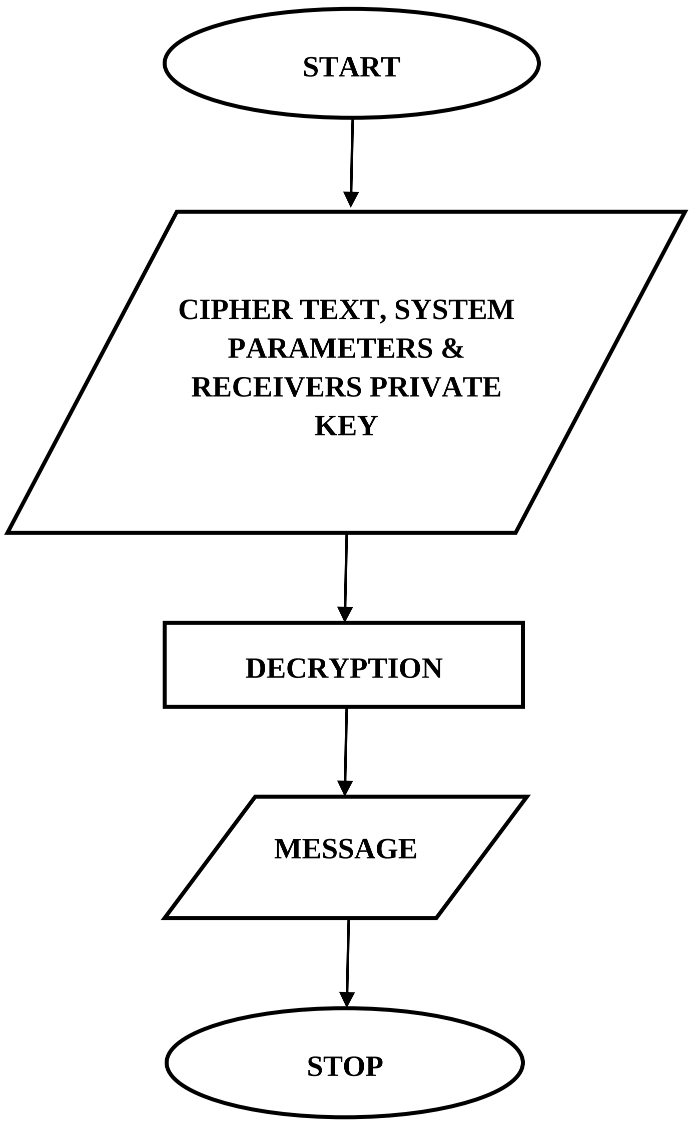


Figure 11: Decryption Algorithm

<u>Flowchart for Signature Algorithm</u>:

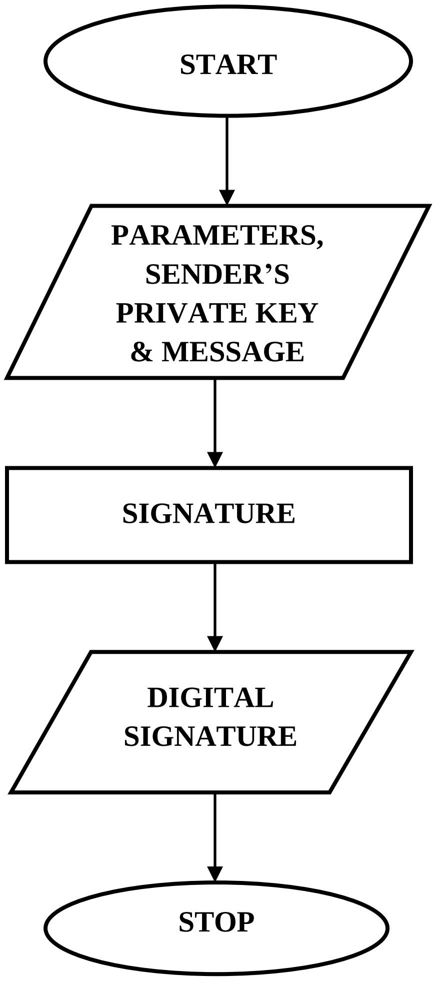


Figure 12: Signature Algorithm

<u>Flowchart for Verification Algorithm</u>:

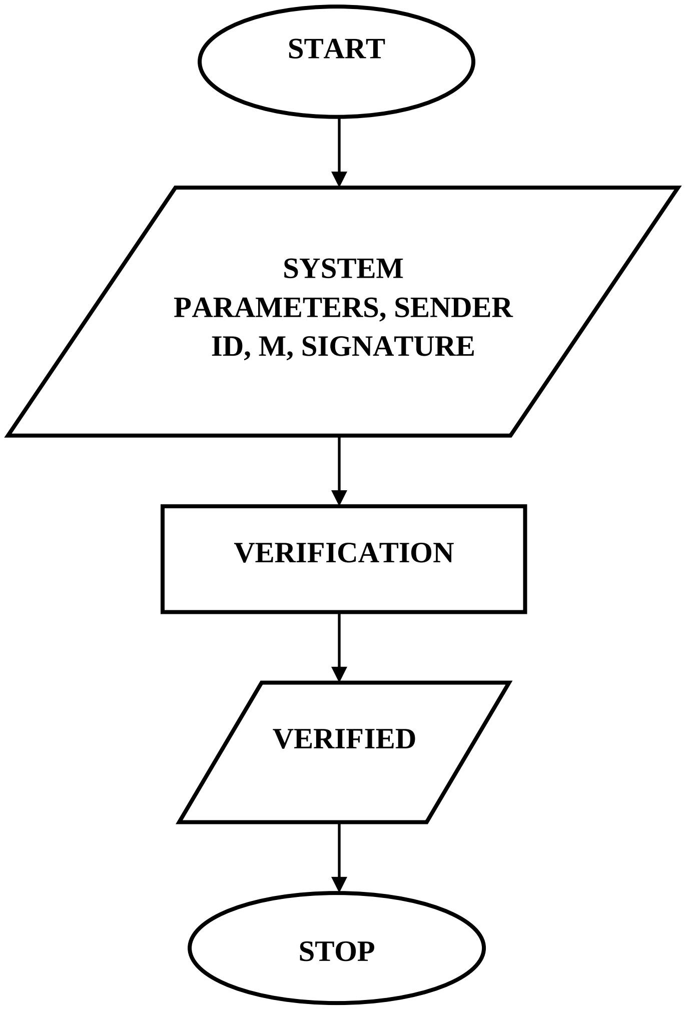


Figure 13: Verification Algorithm

CHAPTER – 4

IMPLEMENTATION DETAIL AND RESULTS

1) SETUP ALGORITHM:

Begin

//Setup algorithm

- ✓ PKG create a master key K_m and the system parameters P.
- ✓ K_m is kept secret and used to generate private key for users.
- ✓ System parameters P are made public for all the users and can be used to generate users' public key with their identities.

End

2) EXTRACT ALGORITHM:

Begin

//Extract algorithm

- ✓ When a user requests his private key from the PKG.
- ✓ PKG will use the identity of this user, system parameters P and master key K_m to generate a private key for this user.

End

3) ENCRYPTION ALGORITHM:

Begin

//Encryption algorithm

When a user wants to encrypt a message and send to another user, he can use the system parameters P, receiver's identity and the message as input to generate the cipher text.

C = Encryption (parameters, receiver ID, M).
End

4) <u>DECRYPTION ALGORITHM:</u>

Begin
//Decryption algorithm Receiving a cipher text, receiver can use the system parameters P and his private key got from the PKG to decrypt the cipher text. **M = Decryption (parameters, k, C)**, k is the private key of the receiver.
End

Key Generation

Let G_1 and G_2 be two groups of some large prime order q and G_1 is an additive group and G_2 is a multiplicative group, we can call ê an admissible pairing if ê: $G_1 \times G_1 \rightarrow G_2$ have the following properties.

1) Bilinear: For all P, Q $\in G_1$ and a, b$\in Z_q^*$,

$$ê(aP, bQ) = ê(abP, Q) = ê(P, abQ) = ê(aP, Q)^b = ê(P,Q)^{ab}.$$

For any a $\in Z^*_q$ and P $\in G_1$, we write aP as the scalar multiplication of group element P by integer a [5].

2) Non-degenerate: There exits P, Q $\in G_1$, such that ê (P, Q) $\neq 1$.

ê does not send all pairs of points in $G_1 \times G_1$ to the identity in G_2. (Hence, if R is a generator of G_1 then ê(R, R) is a generator of G_2.) [6].

3) Computable: For all P, Q $\in G_1$, there exists an efficient way to calculate ê (P, Q).

An admissible pairing can be generated by using a Weil pairing or a Tate pairing. Here, in the cloud we use two levels PKG, the root PKG is level_0 PKG and the PKGs in the private or public clouds are level_1 PKG's. The root setup can be done as follows:

1) Root PKG generates G_1 , G_2 and an admissible pairing $\hat{e}(aP, bQ) = \hat{e}(P, Q)$ not equal to 1 (G_1 , G_2 , $\hat{e}$, P_0 , Q_0 , H_1 , H_2) $\hat{e}: G_1 \times G_1 \rightarrow G_2$.

2) Root PKG chooses a generator $P_0 \in G_1$, picks a random $s_0 \in Z_q^*$ and set $Q_0 = s_0P_0$.

3) Root PKG also chooses cryptographic hash function $H_1: \{0, 1\}^* \rightarrow G_1$ and $H_2: G_2 \rightarrow \{0, 1\}^n$ for some n which denotes the length of a plaintext.

Then the system parameters are (G_1, G_2, $\hat{e}$, P_0, Q_0, H_1, H_2) and are publicly available, s_0 is the root PKG's secret and is known only by the root PKG.

For the lower level PKGs, users and servers in the cloud, they can use the system parameters and any user's identity to generate its public key. And every user or servers in the cloud can connect the PKGs in their cloud domain to get their private keys.

Pairing functions

An application should first initialize a pairing object. This causes PBC to setup curves, groups and other mathematical miscellany. After that, elements can be initialized and manipulated for cryptographic operations.

Pairings involve three groups of prime order. The PBC library calls them G1, G2, and GT, and calls the order r. The pairing is a bilinear map that takes two elements as input, one from G1 and one from G2, and outputs an element of GT.

The elements of G2 are at least as long as G1; G1 is guaranteed to be the shorter of the two. Sometimes G1 and G2 are the same group (i.e. the pairing is symmetric) so their elements can be mixed freely.

Applying pairings

The function pairing_apply can be called to apply a bilinear map. The order of the inputs is important. The first, which holds the output, must be from the group GT. The second must be from G1, the third from G2. Never mix and match G1, G2, and GT groups from different pairings.

Initializing elements

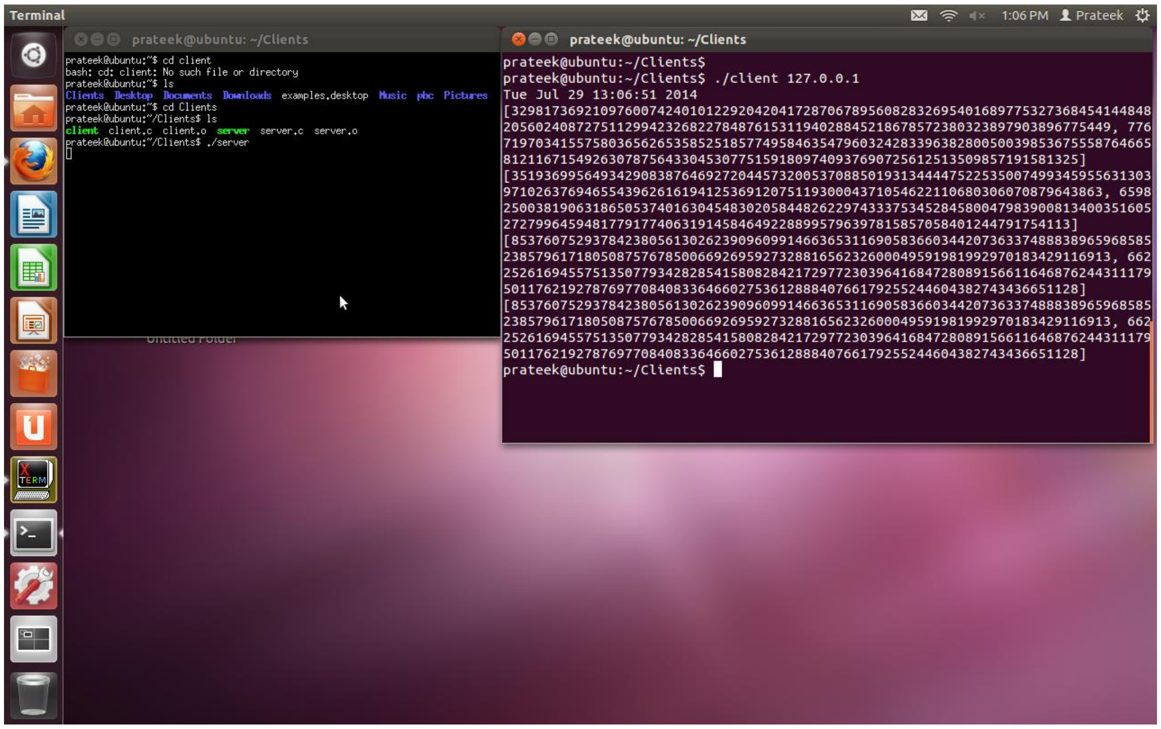
When an element is initialized it is associated with an algebraic structure, such as a particular finite field or elliptic curve group.

We use G1 and G2 to denote the input groups to the pairing, and GT for the output group. All have order r, and Z_r means the ring of integers modulo r. With symmetric pairings, G1 = G2.

Groups, rings, fields

In mathematics, groups, rings and fields should be distinguished, but for implementation, it is simplest lump them together under the same heading. In any event, distinct data types may lead to a false sense of security.

RESULT SCREENSHOT



```
prateek@ubuntu:~$ cd client
bash: cd: client: No such file or directory
prateek@ubuntu:~$ ls
Clients  Desktop  Documents  Downloads  examples.desktop  Music  pbc  Pictures
prateek@ubuntu:~$ cd Clients
prateek@ubuntu:~/Clients$ ls
client  client.c  client.o  server  server.c  server.o
prateek@ubuntu:~/Clients$ ./server
```

```
prateek@ubuntu:~/Clients$
prateek@ubuntu:~/Clients$ ./client 127.0.0.1
Tue Jul 29 13:06:51 2014
[3298173692109760074240101229204204172870678956082832695401689775327368454144848
205602408727511299423268227848761531194028845218678572380323897903896775449, 776
71970341557580365626535852518577495846354796032428339638280050039853675558764665
81211671549263078756433045307751591809740937690725612513509857191581325]
[3519369956493429083876469272044573200537088501931344447522535007499345955631303
9710263769465543962616194125369120751193000437105462211068030607070879643863, 6598
250038190631865053740163045483020584482622297433375345284580047983900813400351605
2727996459481779177406319145846492288995796397815857058401244791754113]
[8537607529378423805613026239096099146636531169058366034420736337488838965968585
238579617180508757678500669269592732881656232600049591981992970183429116913, 662
25261694557513507793428285415808284217297723039641684728089156611646876244311179
50117621927876977084083364660275361288840766179255244604382743436651128]
[8537607529378423805613026239096099146636531169058366034420736337488838965968585
238579617180508757678500669269592732881656232600049591981992970183429116913, 662
25261694557513507793428285415808284217297723039641684728089156611646876244311179
50117621927876977084083364660275361288840766179255244604382743436651128]
prateek@ubuntu:~/Clients$
```

Pairing is generated using Client – Server model

CHAPTER – 5

TESTING

TESTING

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

Verification

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

Validation

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

Type of Test	Will Test be performed?	Comments/Explanation	Software Components
Requirements Testing	Yes	This is done to find what the project is going to do and what type of algorithm is going to be used for the application and which platform or language to be used for implementation	a) C Language
Unit testing	Yes	This would be done to check the independent functionality of each module and to find whether they are producing the desired output or not	a) Calculate G1 b) Calculate G2
Integration	Yes	This would be done to check whether all the modules on integration are performing or	a) Encrypts the message

		not.	
Performance	Yes	This would be done to compare the performance of proposed algorithm to the existing algorithms.	Proposed application with existing algorithms to find whether its efficiency, computational time is better or not.
Security	Yes	This would be done to check whether the system maintains the functionality as intended.	Overall System
Load	Yes	This would be done to find the maximum operating capacity of the proposed system.	Overall system
Volume	Yes	This would be done to test the performance of the system with the size of data the system is to be tested with.	Overall System Dataset

Table 5.1 (Testing Plan)

Test Environment

Test Environment-	
Software Components-	
• **Front End**	**UNIX Operating System**
• **Back End**	**Not required**
• **Operating System**	**UNIX**
• **Other Software required**	**GMP**
Hardware Requirement-	
• **Processor**	**CORE i3**
• **Memory**	**4 GB DDR3 RAM**
• **Monitor**	**15.6” COLOR**
• **Hard Disk**	**1 TB HDD**
• **Other Hardware required**	**No**

Table 5.2 (Testing Environment)

CHAPTER – 6

FINDINGS & CONCLUSION

FINDINGS

I choose C because:

- GMP, which PBC requires is also written in C.
- PBC is intended to be a low-level portable cryptographic library.
- No library dependencies (except standard C libraries),
- Used an existing library GMP because the library's focus is on precision arithmetic and nothing else, and it aims to be as fast as possible on many platforms.
- Another important factor is that GMP is released under a free license.
- GMP is written to deal with extremely large numbers.
- GMP's method for eliminating the need for & and * operators most of the time by declaring a typedef on arrays of size 1.
- GMP is the only library dependency.

CONCLUSION

The quick development of cloud computing bring some security problems as well as many benefits to Internet users. Current solutions have some disadvantages in key management and authentication especially in a hybrid cloud with several Public /private clouds.

- We implemented the algorithms of identity-based cryptography and depicted the principles of hierarchical identity-based cryptography and find the properties of HIBC fit well with the security demands of cloud.

- We proposed to use federated identity management in the cloud and depicted how can the system generate and distribute the public and private keys to users and servers.

- Compared with the current Ws-Security approach, we can see our approach has its advantages in simplifying public key distribution.
- Also we showed how the users and servers in the cloud can generate secret session key without message exchange and authenticate each other with a simple way using identity-based cryptography.
- Also we can see the key escrow problem of identity-based cryptography can be restricted with HIBC approach.

FUTURE SCOPE

I have plan to implement hierarchical identity-based cryptography and show how the users and servers in the cloud can generate secret session key without message exchange and authenticate each other with a simple way using identity-based cryptography.

Using HIBC in a cloud, any user and server can get its own private key from its domain PKG and can calculate the public key of any other party in the cloud knowing its identity. Then it is easy for a sender to add a digital signature using its private key and for a receiver to verify a digital signature using the sender’s public key.

Furthermore one can use the ‘identifier string’ combined with something like a date/time and encrypt messages.

REFERENCES

[1] Gentry, C., Silverberg, A.: Hierarchical ID-Based cryptography. In: Zhen Y.(ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)

[2] Bhaskar Prasad Rimal, Eunmi Choi, "A taxonomy and survey of cloud computing systems", Fifth International Joint Conference on INC, IMS and IDC, published by IEEE Computer Society (2009)

[3] Beak, J., Newmarch, J., Safavi-Naini, R., Susilo, W.: A Survey of Identity-Based Cryptography.In: Proc. of the 10th Annual Conference for Australian UNIX User's Group (AUUG2004), pp. 95–102 (2004)

[4] Horwitz, J., Lynn, B.: Toward Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481.
Springer, Heidelberg (2002)

[5] Hoon Wei Lim and Kenneth G. Paterson: "Identity-Based Cryptography for Grid Security" Information Security Group Royal Holloway, University of London Egham, Surrey TW20 0EX, UK

[6] Joonsang Baek, Jan Newmarch, Reihaneh Safavi-Naini and Willy Susilo
A Survey of Identity-Based Cryptography

[7] Mao, W.: An Identity-based Non-interactive Authentication Framework for Computational Grids. HP Lab, Technical Report HPL-2004-96 (June 2004)

[8] Y.Jadeja, K.Modi: "Cloud Computing - Concepts, Architecture and Challenges" International Conference on Computing, Electronics and Electrical Technologies [ICCEET] (2012)

[9] Ramgovind S, Eloff MM: The Management of Security in Cloud Computing, IEEE (2010)

[10] http://en.wikipedia.orglwikilCloud_computing

[11] Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)

[12] Crampton, J., Lim, H.W., Paterson, K.G.: What Can Identity-Based Cryptography Offer to Web Services? In: Proceedings of the 5th ACM Workshop on Secure Web Services (SWS 2007), Alexandria, Virginia, USA, pp. 26–36. ACM Press, New York (2007)

[13] Chappell, D.: A Short Introduction to Cloud Platforms,
http://www.davidchappell.com/CloudPlatforms–Chappell.pdf

[14] Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: Proceedings of the 2000 Symposium on Cryptography and Information Security, Okinawa, Japan (January 2000)

[15] Peeyush Mathur, Nikhil Nishchal, "Cloud Computing: New challenge to the entire computer industry", 2010 1st International Conference on Parallel, Distributed and Grid Computing (PDGC - 2010).