

## CHAPTER 1

### INTRODUCTION

The term “cloud computing” is a recent buzzword in the IT world. Behind this fancy poetic phrase there lies a true picture of the future of computing for both in technical perspective and social perspective. Though the term “Cloud Computing” is recent but the idea of centralizing computation and storage in distributed data centers maintained by third party companies is not new but it came in way back in 1990s along with distributed computing approaches like grid computing. Cloud computing is aimed at providing IT as a service to the cloud users on-demand basis with greater flexibility, availability, reliability and scalability with utility computing model.

So the birth of cloud computing is very recent phenomena although its root belongs to some old ideas with new business, technical and social perspectives. From the architectural point of view cloud is naturally build on an existing grid based architecture and uses the grid services and adds some technologies like virtualization and some business models. In brief cloud is essentially a bunch of commodity computers networked together in same or different geographical locations, operating together to serve a number of customers with different need and workload on demand basis with the help of virtualization.

Cloud Computing provides us a means by which we can access the applications as utilities, over the Internet. It allows us to create, configure, and customize applications online. The term Cloud refers to a Network or Internet. Cloud Computing refers to manipulating, configuring, and accessing the applications online. It offers online data storage, infrastructure and application. Cloud can provide services over network, i.e., on public networks or on private networks, i.e., WAN, LAN or VPN. Applications such as e-mail, web conferencing, customer relationship management (CRM), all run in cloud. Basic Concepts There are certain services and models working behind the scene making the cloud computing feasible and accessible to end users.

Following are the working models for cloud computing:

- Deployment Models
- Service Models

## **BLOCKCHAIN :**

Blockchain is a fiery research and application field. It is an encrypted distributed anonymous ledger system recording transactions between participants, which solves inconsistent information consensus in a distributed network, such as the Byzantine Generals problem. Based on the blockchain platform, smart contracts have realized decentralization, automatic execution, and other functions, enriching the research and application of the blockchain field—for example, cryptocurrency combined with blockchain technology, Bitcoin, and Ethereum.

## **BLOCK CHAIN ALGORIHM :**

A block chain is a growing list of records, called blocks, that are linked together using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. The timestamp proves that the transaction data existed when the block was published in order to get into its hash. As blocks each contain information about the block previous to it, they form a chain, with each additional block reinforcing the ones before it. Therefore, block chains are resistant to modification of their data because once recorded, the data in any given block cannot be altered retroactively without altering all subsequent blocks.

Block chains are typically managed by a peer-to-peer network for use as a publicly distributed ledger, where nodes collectively adhere to a protocol to communicate and validate new blocks. Although block chain records are not unalterable as forks are possible, block chains may be considered secure by design and exemplify a distributed computing system with high Byzantine fault tolerance.

### **1.1 AIM AND OBJECTIVE :**

The main objective is to provide the cloud security through the various block chain encryption and decryption algorithm.

### **1.2 ORGANIZATION OF THE PROJECT :**

**Chapter 1 :** Introduction, Block Chain Algorithm, deployment models, Service models, Infrastructure as a service, PaaS, SaaS, Cloud Security, Uses of Cloud Computing.

**Chapter 2 :** Literature Review, Towards Trusted cloud computing, Seeding cloud with trust anchor, Domain based storage protection with Secure Acces Control for the Cloud, Security Aspects of e-Health System Migration to the Cloud, Securely Launching Virtual Machine on Trustworthy Platform, Secure and Efficient Access to Outsourced Data, Policy Sealed Data, Cipher Text, Fuzzy Identity Based Encryption, On the Security of Public Key Protocols.

**Chapter 3 :** System Analysis, Existing System, drawbacks, Proposed System, Advantage

**Chapter 4 :** DH Algorithm, AES Algorithm, RSA Algorithm, ECC Algorithm

**Chapter 5 :** System Model, Registration And Encryption, Database Storage, Group Key Generation, Sharing Data Within the Group.

**Chapter 6 :** Result and Output, contains all output screenshot related to our project work.

**Chapter 7 :** Gives us the conclusion and references of the project.

**Appendices :** Softwere and Hardware Description, Source Code of the Project and References.

### **1.3 DEPLOYMENT MODELS**

Deployment models define the type of access to the cloud can have any of the four types of access: Public, Private, Hybrid and Community. The Public Cloud allows systems and services to be easily accessible to the general public. Public cloud may be less secure because of its openness, e.g., e-mail. The Private Cloud allows systems and services to be accessible within an organization. It offers increased security because of its private nature. The Community Cloud allows systems and services to be accessible by group of organizations. The Hybrid Cloud is mixture of public and private cloud. However, the critical activities are performed using private cloud while the non-critical activities are performed using public cloud.

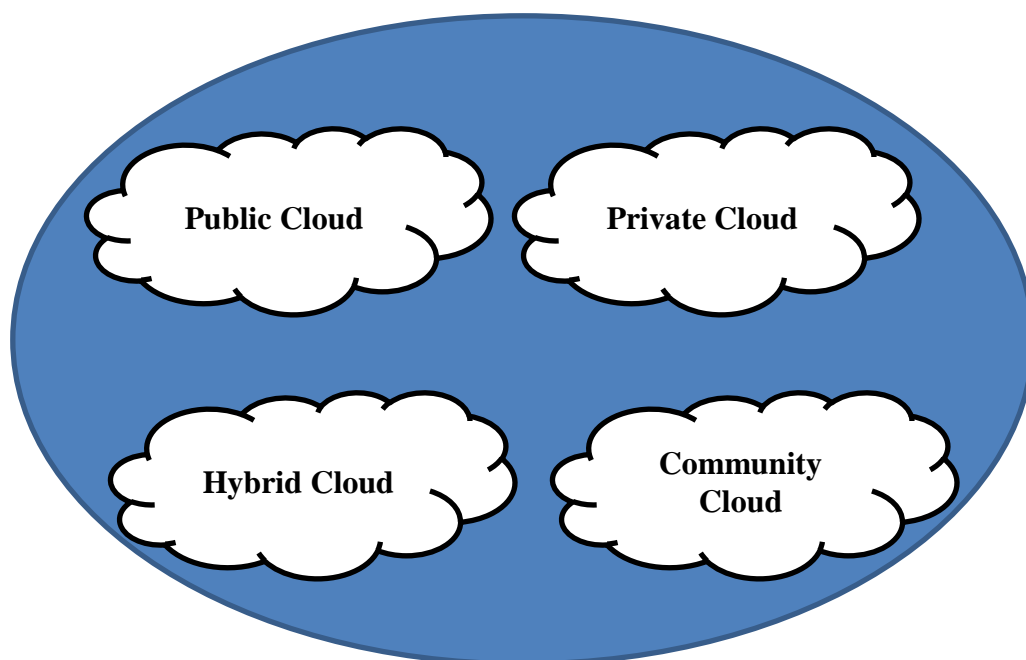


Figure:1.1 Deployment Model

## 1.4 SERVICE MODELS

Service Models are the reference models on which the Cloud Computing is based. These can be categorized into three basic service models as listed below:

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

There are many other service models all of which can take the form like XaaS, i.e., Anything as a Service. This can be Network as a Service, Business as a Service, Identity as a Service, Database as a Service or Strategy as a Service. The Infrastructure as a Service (IaaS) is the most basic level of service. Each of the service models make use of the underlying service model, i.e., each inherits the security and management mechanism from the underlying model.

### 1.4.1 INFRASTRUCTURE AS A SERVICE (IaaS)

IaaS provides access to fundamental resources such as physical machines, virtual machines, virtual storage, etc. In an IaaS model, a third-party provider hosts hardware, software, servers, storage and other infrastructure components on behalf of its users. IaaS providers also host users'

applications and handle tasks including system maintenance, backup and resiliency planning. IaaS platforms offer highly scalable resources that can be adjusted on-demand. This makes IaaS well-suited for workloads that are temporary, experimental or change unexpectedly. Other characteristics of IaaS environments include the automation of administrative tasks, dynamic scaling, desktop virtualization and policy-based services.

IaaS customers pay on a per-use basis, typically by the hour, week or month. Some providers also charge customers based on the amount of virtual machine space they use. This pay-as-you-go model eliminates the capital expense of deploying in-house hardware and software. However, users should monitor their IaaS environments closely to avoid being charged for unauthorized services. Because IaaS providers own the infrastructure, systems management and monitoring may become more difficult for users. Also, if an IaaS provider experiences downtime, users' workloads may be affected. For example, if a business is developing a new software product, it might be more cost-effective to host and test the application through an IaaS provider. Once the new software is tested and refined, it can be removed from the IaaS environment for a more traditional in-house deployment or to save money or free the resources for other projects. Leading IaaS providers include Amazon Web Services (AWS), Windows Azure, Google Compute Engine, Rackspace Open Cloud, and IBM Smart Cloud Enterprise.

#### **1.4.2 PLATFORM AS A SERVICE (PaaS)**

PaaS provides the runtime environment for applications, development & deployment tools, etc. In a PaaS model, a cloud provider delivers hardware and software tools usually those needed for application development -- to its users as a service. A PaaS provider hosts the hardware and software on its own infrastructure. As a result, PaaS frees users from having to install in-house hardware and software to develop or run a new application.

PaaS does not typically replace a business' entire infrastructure. Instead, a business relies on PaaS providers for key services, such as Java development or application hosting. For example, deploying a typical business tool locally might require an IT team to buy and install hardware, operating systems, middleware (such as databases, Web servers and so on) the actual application, define user access or security, and then add the application to existing systems management or application performance monitoring (APM) tools. IT teams must then maintain all of these

resources over time. A PaaS provider, however, supports all the underlying computing and software; users only need to log in and start using the platform – usually through a Web browser interface. Most PaaS platforms are geared toward software development, and they offer developers several advantages. For example, PaaS allows developers to frequently change or upgrade operating system features. It also helps development teams collaborate on projects.

Users typically access PaaS through a Web browser. PaaS providers then charge for that access on a per-use basis. Some PaaS providers charge a flat monthly fee to access the platform and the apps hosted within it. It is important to discuss pricing, service uptime and support with a PaaS provider before engaging their services. Since users rely on a provider's infrastructure and software, vendor lock-in can be an issue in PaaS environments. Other risks associated with PaaS are provider downtime or a provider changing its development roadmap. If a provider stops supporting a certain programming language, users may be forced to change their programming language, or the provider itself. Both are difficult and disruptive steps. Common PaaS vendors include Salesforce.com's Force.com, which provides an enterprise customer relationship management (CRM) platform. PaaS platforms for software development and management include Appian, Mendix, Amazon Web Services (AWS) Elastic Beanstalk, Google App Engine and Heroku.

### 1.4.3 SOFTWARE AS A SERVICE (SaaS)

SaaS model allows to use software applications as a service to end users. SaaS removes the need for organizations to install and run applications on their own computers or in their own data centers. This eliminates the expense of hardware acquisition, provisioning and maintenance, as well as software licensing, installation and support. Other benefits of the SaaS model include:

**Flexible payments:** Rather than purchasing software to install, or additional hardware to support it, customers subscribe to a SaaS offering. Generally, they pay for this service on a monthly basis using a pay-as-you-go model. Transitioning costs to a recurring operating expense allows many businesses to exercise better and more predictable budgeting. Users can also terminate SaaS offerings at any time to stop those recurring costs.

**Scalable usage:** Cloud services like SaaS offer high scalability, which gives customers the option to access more, or fewer, services or features on-demand.

**Automatic updates:** Rather than purchasing new software, customers can rely on a SaaS provider to automatically perform updates and patch management. This further reduces the burden on in-house IT staff.

**Accessibility and persistence:** Since SaaS applications are delivered over the internet, users can access them from any Internet-enabled device and location.

Cloud computing has progressed from a bold vision to massive deployments in various application domains. However, the complexity of technology underlying cloud computing introduces novel security risks and challenges. Threats and mitigation techniques for the IaaS model have been under intensive scrutiny in recent years, while the industry has invested in enhanced security solutions and issued best practice recommendations.

From an end user point of view the security of cloud infrastructure implies unquestionable trust in the cloud provider, in some cases corroborated by reports of external auditors. While providers may offer security enhancements such as protection of data at rest, end-users have limited or no control over such mechanisms. There is a clear need for usable and cost-effective cloud platform security mechanisms suitable for organizations that rely on cloud infrastructure. One such mechanism is platform integrity verification for compute hosts that support the virtualized cloud infrastructure.

Several large cloud vendors have signaled practical implementations of this mechanism, primarily to protect the cloud infrastructure from insider threats and advanced persistent threats. We see two major improvement vectors regarding these implementations. First, details of such proprietary solutions are not disclosed and can thus not be implemented and improved by other cloud platforms. Second, to the best of our knowledge, none of the solutions provides cloud tenants a proof regarding the integrity of compute hosts supporting their slice of the cloud infrastructure.

To address, propose a set of protocols for trusted launch of virtual machines (VM) in IaaS, which provide tenants with a proof that the requested VM instances were launched on a host with an expected software stack. Another relevant security mechanism is encryption of virtual disk volumes, implemented and enforced at compute host level. While support data encryption at rest is

offered by several cloud providers and can be configured by tenants in their VM instances, functionality and migration capabilities of such solutions are severely restricted.

In most cases cloud providers maintain and manage the keys necessary for encryption and decryption of data at rest. This further convolutes the already complex data migration procedure between different cloud providers, disadvantaging tenants through a new variation of vendor lock-in. Tenants can choose to encrypt data on the operating system (OS) level within their VM environments and manage the encryption keys themselves. However, this approach suffers from several drawbacks.

First, the underlying compute host will still have access encryption keys whenever the VM performs cryptographic operations second, this shifts towards the tenant the burden of maintaining the encryption software in all their VM instances and increases the attack surface; third, this requires injecting, migrating and later securely withdrawing encryption keys to each of the VM instances with access to the encrypted data, increasing the probability than an attacker eventually obtains the keys.

DBSP, a virtual disk encryption mechanism where encryption of data is done directly on the compute host, while the key material necessary for re-generating encryption keys is stored in the volume metadata. This approach allows easy migration of encrypted data volumes and withdraws the control of the cloud provider over disk encryption keys.

DBSP significantly reduces the risk of exposing encryption keys and keeps a low maintenance overhead for the tenant in the same time providing additional control over the choice of the compute host based on its software stack. We focus on the Infrastructure-as-a-Service model in a simplified form, it exposes to its tenants a coherent platform supported by compute hosts which operate VM guests that communicate through a virtual network. The system model while migrating a currently deployed, distributed electronic health record (EHR) system to an IaaS platform.

The data and operational security in IaaS securing our digital assets has become increasingly challenging as our reliance on rapidly evolving technologies continues to grow. The security perimeter in computing has changed from a well define boundary that was relatively easy to identify and defend, to an elastic boundary that is constantly changing and for which the threats are



constantly evolving. This paper investigates the complex security challenges that are introduced by the trend towards Infrastructure as a Service (IaaS) based cloud computing. While not exhaustive, it identifies some technological and legal issues and concerns from the perspectives of identified stakeholders, and suggests some future directions for security research and development to help advance the security posture of this technology.

## 1.5 CLOUD SECURITY :

While security and privacy concerns are similar across cloud services and traditional non-cloud services, those concerns are amplified by the existence of external control over organizational assets and the potential for mismanagement of those assets. Transitioning to public cloud computing involves a transfer of responsibility and control to the cloud provider over information as well as system components that were previously under the customer's direct control. Despite this inherent loss of control, the cloud service customer still needs to take responsibility for its use of cloud computing services in order to maintain situational awareness, weigh alternatives, set priorities, and effect changes in security and privacy that are in the best interest of the organization. Transitioning to public cloud computing involves a transfer of responsibility and control to the cloud provider over information as well as system components that were previously under the customer's direct control. The customer achieves this by ensuring that the contract with the provider and its associated cloud service agreement has appropriate provisions for security and privacy. In particular, the agreement must help maintain legal protections for the privacy of data stored and processed on the provider's systems. The customer must also ensure appropriate integration of cloud computing services with their own systems for managing security and privacy.

There are a number of security risks associated with cloud computing that must be adequately addressed :

- **Loss of governance:** In a public cloud deployment, customers code control to the cloud provider over a number of issues that may affect security. Yet cloud service agreements may not offer a commitment to resolve such issues on the part of the cloud provider, thus leaving gaps in security defenses.

- **Responsibility ambiguity:** Responsibility over aspects of security may be split between the provider and the customer, with the potential for vital parts of the defenses to be left unguarded

if there is a failure to allocate responsibility clearly. This split is likely to vary depending on the cloud computing model used (e.g., IaaS vs. SaaS).

- **Authentication and Authorization:** The fact that sensitive cloud resources are accessed from anywhere on the Internet heightens the need to establish with certainty the identity of a user - especially if users now include employees, contractors, partners and customers. Strong authentication and authorization becomes a critical concern.

- **Isolation failure:** Multi-tenancy and shared resources are defining characteristics of public cloud computing. This risk category covers the failure of mechanisms separating the usage of storage, memory, routing and even reputation between tenants (e.g. so-called guest-hopping attacks).

- **Compliance and legal risks:** The cloud customer's investment in achieving certification (e.g., to demonstrate compliance with industry standards or regulatory requirements) may be lost if the cloud provider cannot provide evidence of their own compliance with the relevant requirements, or does not permit audits by the cloud customer. The customer must check that the cloud provider has appropriate certifications in place.

## 1.6. USES OF CLOUD COMPUTING :

The user probably using cloud computing right now, even if you don't realize it. If you use an online service to send email, edit documents, watch movies or TV, listen to music, play games or store pictures and other files, it is likely that cloud computing is making it all possible behind the scenes. The first cloud computing services are barely a decade old, but already a variety of organization from tiny startups to global corporations, government agencies to non-profits are embracing the technology for all sorts of reasons. Here are a few of the things you can do with the cloud:

- Create new apps and services
- Store, back up and recover data
- Host websites and blogs
- Stream audio and video
- Deliver software on demand
- Analyse data for patterns and make predictions

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Towards Trusted Cloud Computing :**

Nuno Santos, Krishna P. Gummadi and Rodrigo Rodrigues [13] (2009) propose Cloud computing infrastructures enable companies to cut costs by outsourcing computations on-demand. However, clients of cloud computing services currently have no means of verifying the confidentiality and integrity of their data and computation. To address this problem to propose the design of a trusted cloud computing platform (TCCP). TCCP enables Infrastructure as a Service (IaaS) providers such as Amazon EC2 to provide a closed box execution environment that guarantees confidential execution of guest virtual machines.

Trusted cloud computing platform (TCCP) for ensuring the confidentiality and integrity of computations that are outsourced to IaaS services. The TCCP provides the abstraction of a closed box execution environment for a customer's VM, guaranteeing that no cloud provider's privileged administrator can inspect or tamper with its content. Moreover, before requesting the service to launch a VM, the TCCP allows a customer to reliably and remotely determine whether the service backend is running a trusted TCCP implementation. This capability extends the notion of attestation to the entire service, and thus allows a customer to verify if its computation will run securely. In the proposed system, show how to leverage the advances of trusted computing technologies to design the TCCP.

#### **2.2 Seeding Clouds with Trust Anchors :**

Joshua Schiffman and his co-authors [15] (2010) proposes the paper for the customers security critical data processing needs are beginning to push back strongly against using cloud computing. Cloud vendors run their computations upon cloud provided VM systems, but customers are worried such host systems may not be able to protect themselves from attack, ensure isolation of customer processing, or load customer processing correctly. To provide assurance of data processing protection in clouds to customers, user advocate methods to improve cloud transparency using hardware-based attestation mechanisms.

The centralized management of cloud data centers is ideal for attestation frameworks, enabling the development of a practical approach for customers to trust in the cloud platform. Specifically, propose a cloud verifier service that generates integrity proofs for customers to verify the integrity and access control enforcement abilities of the cloud platform that protect the integrity of customer's application VMs in IaaS clouds. While a cloud-wide verifier service could present a significant system bottleneck, demonstrate that aggregating proofs enables significant overhead reductions. As a result, transparency of data security protection can be verified at cloud-scale.

The main three challenges has been discussed are that cloud providers face when generating proofs that can placate a user's concerns: First that cloud vendors provide a proof of data security protection of their hosts and customer processing; Second proofs have a clear meaning to cloud customers; and Third proofs can be generated effectively and efficiently in a cloud computing environment.

### **2.3 Domain Based Storage Protection with Secure Access Control for the Cloud :**

Nicolae Paladi, Antonis Michalas and Christian Gehrman [10] (2014) proposes cloud computing has evolved from a promising concept to one of the fastest growing segments of the IT industry. However, many businesses and individuals continue to view cloud computing as a technology that risks exposing their data to unauthorized users. To introduce a data confidentiality and integrity protection mechanism for Infrastructure as a Service (IaaS) clouds, which relies on trusted computing principles to provide transparent storage isolation between IaaS clients. The system also address the absence of reliable data sharing mechanisms, by providing an XML-based language framework which enables clients of IaaS clouds to securely share data and clearly denies access rights granted to peers. The proposed improvements have been prototyped as a code extension for a popular cloud platform. Full-disk encryption has emerged as a solid solution for data confidentiality protection and is also mentioned in as a solution to the "dirty disks" problem. However, fulldisk encryption creates hurdles for data sharing, widely recognized as an essential feature for cloud applications. Despite the variety of available open source cloud management platforms (e.g OpenStack, Eucalyptus, OpenNebula), allocation of read-write permissions for shared data between collaborating tenants still remains an open problem. The system improve and

extend previous work by adding capabilities to both grant access to data to other IaaS cloud clients and assign access permissions.

## **2.4 Security Aspects of e-Health Systems Migration to the Cloud :**

Antonis Michalas, N.Paladi, and C.Gehrmann [7] (2014) proposed As adoption of e-health solutions advances, new computing paradigms such as cloud computing bring the potential to improve efficiency in managing medical health records and help reduce costs. However, these opportunities introduce new security risks which cannot be ignored. Based on our experience with deploying part of the Swedish electronic health records management system in an infrastructure cloud, we make an overview of major requirements that must be considered when migrating e-health systems to the cloud. Furthermore, in depth a new attack vector inherent to cloud deployments and present a novel data confidentiality and integrity protection mechanism for infrastructure clouds. This contribution aims to encourage exchange of best practices and lessons learned in migrating public e-health systems to the cloud.

Visions of an electronic healthcare system are more than twenty years old. Researchers aimed for a paperless medical system where patients and doctors are able to book appointments via the Internet, create electronic prescriptions and store their medical history in a central database, easily accessible from anyone with appropriate access rights. During these years, there has been a steady increase in research focus and funding aiming to modernize existing healthcare systems and provide reliable and cost effective e-health services. Both private organizations, such as Microsoft, Google and IBM, and public administration bodies have taken steps towards e-health.

For example, president of the United States B. Obama, approved \$38 billion to digitize the American health care and believes that at the end of 2014 the nation's health records will be fully computerized. In addition, the Australian government invested \$20.3 million in "telehealth" projects, Tasmania committed \$1.8 million in order to update the information systems responsible for four of its public hospitals, while Germany has introduced the electronic health card a challenging mission in which all insured Germans received a smart card with which they can securely communicate with various healthcare stakeholders (doctors, hospitals, pharmacies etc) by means of telematics.

## 2.5 Securely Launching Virtual Machines on Trustworthy Platforms in a Public Cloud

Mudassar Aslam, L.Rasmusson, and M. Björkman [1],(2012) proposed the Infrastructure-as-a-Service (IaaS) cloud model which allows cloud users to run their own virtual machines (VMs) on available cloud computing resources. IaaS gives enterprises the possibility to outsource their process workloads with minimal effort and expense. However, one major problem with existing approaches of cloud leasing, is that the users can only get contractual guarantees regarding the integrity of the offered platforms. The fact that the IaaS user himself or herself cannot verify the providerpromisedcloud platform integrity, is a security risk which threatens to prevent the IaaS business in general. The author address this issue and propose a novel secure VM launch protocol using Trusted Computing techniques. VM launch protocol allows the cloud IaaS users to securely bind the VM to a trusted computer configuration such that the clear text VM only will run on a platform that has been booted into a trustworthy state. The capability builds user confidence and can serve as an important enabler for creating trust in public clouds. To evaluate the feasibility of our proposed protocol via a full scale system implementation and perform a system security analysis. IaaS gives enterprises the possibility to outsource their process workloads with minimal effort.

A small company without security skills or an ordinary IT service consumer might trust a public cloudservice provider and in some cases prefer cloud services over self-hosted services with a belief that his other cloud provider can offer better security by recruiting specialized staff and equipment. In contrast, most large or medium size enterprises have higher security requirements for their own or their business users sensitive data; and if their data is compromised due to a security breach in the cloud provider network, it will results in serious legal and business setbacks. Therefore, these enterprises are reluctant to host their services in a public cloud unless they get trusted ways to validate the contractual security guarantees provided by the cloud provider.

The focus of our work is to introduce technical ways to verify the security guarantees provided by the cloud service provider. To achieve this by allowing the cloud user to cryptographically bind the user virtual machine (VM) to a trustworthy state of the provisioned cloud platform. Furthermore, to ensure that the whole launch process meets all expected major security requirements of a high quality public service with respect to authentication and secure transfer.

According to suggested VM launch protocol, a particular VM is not even sent to the provider network if no platform with the expected security guarantees can be offered by the IaaS cloud.

## **2.6 Secure and Efficient Access to Outsourced Data :**

Weichao Wang [17] (2009) et al proposes to provide secure and efficient access to large scale outsourced data is an important component of cloud computing. In the paper, propose a mechanism to solve this problem in owner-write-users-read applications. The system propose to encrypt every data block with a different key so that flexible cryptography-based access control can be achieved. Through the adoption of key derivation methods, the owner needs to maintain only a few secrets. Analysis shows that the key derivation procedure using hash functions will introduce very limited computation overhead. The system propose to use over-encryption and/or lazy revocation to prevent revoked users from getting access to updated data blocks. The system design mechanisms to handle both updates to outsourced data and changes in user access rights. To investigate the overhead and safety of the proposed approach, and study mechanisms to improve data access efficiency.

The system focus on the data outsourcing scenario, the data can be updated only by the original owner. At the same time, end users with different access rights need to read the information in an efficient and secure way. Both data and user dynamics must be properly handled to preserve the performance and safety of the outsourced storage system. Before presenting the details of the proposed approach, the system use an example to illustrate the potential applications. The world's largest collider accelerator LHC can generate about 10 PB (Peta-Bytes,  $10^{15}$  bytes) data each year.

The data can be stored on a third party server and the European Organization for Nuclear Research (CERN) may publish new data, update existing records, and delete expired information. The data can be accessed by scientists in different countries. Since the scientists may have different security clearance levels, encryption based access control will be adopted. At the same time, methods must be designed to support dynamic changes of the access rights of end users.

## 2.7 Versatile Key Management for Secure Cloud Storage :

Sebastian Graf [4] (2012) et al Not only does storing data in the cloud utilize specialized infrastructures facilitating immense scalability and high availability, but it also offers a convenient way to share any information with user-defined third-parties. However, storing data on the infrastructure of commercial third party providers, demands trust and confidence. Simple approaches, like merely encrypting the data by providing encryption keys, which at most consist of a shared secret supporting rudimentary data sharing, do not support evolving sets of accessing clients to common data. Based on approaches from the area of stream-encryption, author proposes an adaption for enabling scalable and flexible key management within heterogeneous environments like cloud scenarios. Representing access-rights as a graph, we distinguish between the keys used for encrypting hierarchical data and the encrypted updates on the keys enabling flexible join leave operations of clients. This distinction allows us to utilize the high availability of the cloud as updating mechanism without harming confidentiality.

Graph-based key management results in an adaption of nodes related to the changed key. The updates on the keys again continuously create an overhead related to the number of these updated nodes. The proposed scalable approach utilizes cloud-based infrastructures for confidential data and key sharing in collaborative workflows supporting variable client-sets. The author approach consists of three components, namely a global Key Graph stored on a trusted third party environment, encrypted updates on the Key Graph as well as versioned data, both stored encrypted in the cloud. The Key Graph relies on existing graph-based key management approaches namely Versa Key extended as a DAG. This approach binds key material to nodes related to each other representing the DAG where the source nodes (represented by the “Client Keys”(CKs) with the Key Graph) constitute the client rights and the terminal nodes represent the most common access rights called “Encryption Keys”(EKs).

## 2.8 Policy-Sealed Data: A New Abstraction for Building Trusted Cloud Services :

Nuno Santos [14] (2012) et al propose accidental or intentional mismanagement of cloud software by administrators poses a serious threat to the integrity and confidentiality of customer data hosted by cloud services. Trusted computing provides an important foundation for designing cloud services that are more resilient to these threats. However, current trusted computing



technology is ill-suited to the cloud as it exposes too many internal details of the cloud infrastructure, hinders fault tolerance and load-balancing flexibility, and performs poorly. The system presents Excalibur, a system that addresses these limitations by enabling the design of trusted cloud services.

Excalibur provides a new trusted computing abstraction, called *policy-sealed data*, that lets data be *sealed* (i.e., encrypted to a customer-defined policy) and then *unsealed* (i.e., decrypted) only by nodes whose configurations match the policy. To provide this abstraction, Excalibur uses attribute-based encryption, which reduces the overhead of key management and improves the performance of the distributed protocols employed. To demonstrate that Excalibur is practical, we incorporated it in the Eucalyptus open-source cloud platform. Policy-sealed data can provide greater confidence to Eucalyptus customers that their data is not being mismanaged.

The system presents Excalibur, a system that provides cloud service designers with new trusted computing abstractions that overcome these barriers. These abstractions provide another critical building block for constructing services that offer better guarantees regarding data integrity, confidentiality, or location. Excalibur provides a new trusted computing abstraction, called policy-sealed data.

Excalibur's design includes two main innovations crucial to overcoming the concerns posed by using TPMs in the cloud. Excalibur provides a new trusted computing abstraction, called policy-sealed data, that allows customer data to be encrypted according to a customer chosen policy and guarantees that only the cloud nodes whose configuration satisfies that policy can decrypt and retrieve the data. Excalibur also implements the policy-sealed data abstraction in a way that overcomes the inefficiency hurdles of current TPMs and scales to the demand of cloud services.

## **2.9 Property Based Attestation for Computing Platforms: Caring About Properties, Not Mechanisms :**

Ahmad Reza Sadeghi [11] (2004) et al proposes over the past years, the computing industry has started various initiatives announced to increase computer security by means of new hardware architectures. The most notable effort is the Trusted Computing Group (TCG) and the Next Generation Secure Computing Base (NGSCB). This technology offers useful new functionalities

as the possibility to verify the integrity of a platform (attestation) or binding quantities on a specific platform (sealing).

The System point out the deficiencies of the attestation and sealing functionalities proposed by the existing specification of the TCG: we show that these mechanisms can be misused to discriminate certain platforms, i.e., their operating systems and consequently the corresponding vendors. A particular problem in this context is that of managing the multitude of possible configuration. Moreover, highlight other shortcomings related to the attestation, namely system updates and backup. Clearly, the consequences caused by these problems lead to an unsatisfactory situation both for the private and business branch, and to an unbalanced market when such platforms are in wide use.

The system propose a completely new approach: the attestation of a platform should not depend on the specific software or/and hardware (configuration) as it is today's practice but only on the properties" that the platform offers. Thus, a property-based attestation should only verify whether these properties are sufficient to fulfill certain (security) requirements of the party who asks for attestation. We propose and discuss a variety of solutions based on the existing Trusted Computing (TC) functionality. The system also demonstrates how a property based attestation protocol can be realized based on the existing TC hardware such as a Trusted Platform Module (TPM).

## **2.10 Cipher text-Policy Attribute-Based Encryption :**

John Bethen [12] (2007) court et al proposes several distributed systems a user should only be able to access data if a user passes a certain set of credentials or attributes. Currently, the only method for enforcing such policies is to employ a trusted server to store the data and mediate access control. However, if any server storing the data is compromised, then the confidentiality of the data will be compromised. In the paper a system for realizing complex access control on encrypted data that we call Cipher text-Policy Attribute Based Encryption. By using the techniques encrypted data can be kept confidential even if the storage server is untrusted; moreover, our methods are secure against collusion attacks.

Previous Attribute Based Encryption systems used attributes to describe the encrypted data and built policies into user's keys; while in our system attributes are used to describe a user's credentials, and a party encrypting data determines a policy for who can decrypt. Thus, our methods are conceptually closer to traditional access control methods such as Role-Based Access Control (RBAC). In addition, the system provide an implementation of our system and give performance measurements.

Traditionally, the type of expressive access control is enforced by employing a trusted server to store data locally. The server is entrusted as a reference monitor that checks that a user presents proper certification before allowing him to access records or files. However, services are increasingly storing data in a distributed fashion across many servers. Replicating data across several locations has advantages in both performance and reliability.

The drawback of this trend is that it is increasingly difficult to guarantee the security of data using traditional methods; when data is stored at several locations, the chances that one of them has been compromised increases dramatically. For these reasons we would like to require that sensitive data is stored in an encrypted form so that it will remain private even if a server is compromised.

### **2.11 Fuzzy Identity-Based Encryption :**

Amit Sahai [12] (2005) et al proposed a new type of Identity-Based Encryption (IBE) scheme that we call Fuzzy Identity-Based Encryption. In Fuzzy IBE we view an identity as set of descriptive attributes. A Fuzzy IBE scheme allows for a private key for an identity,  $!$ , to decrypt a cipher text encrypted with an identity,  $!0$ , if and only if the identities  $!$  and  $!0$  are close to each other as measured by the “set overlap” distance metric. A Fuzzy IBE scheme can be applied to enable encryption using biometric inputs as identities; the error-tolerance property of a Fuzzy IBE scheme is precisely what allows for the use of biometric identities, which inherently will have some noise each time they are sampled.

Additionally, the system show that Fuzzy-IBE can be used for a type of application that we term “attribute-based encryption”. In the paper we present two constructions of Fuzzy IBE schemes. Our constructions can be viewed as an Identity-Based Encryption of a message under several attributes that compose a (fuzzy) identity. The IBE schemes are both error-tolerant and secure

against collusion attacks. Additionally, the basic construction of system does not use random oracles.

The system prove the security of our schemes under the Selective-ID security model. Identity-Based Encryption (IBE) allows for a sender to encrypt a message to an identity without access to a public key certificate. The ability to do public key encryption without certificates has many practical applications. For example, a user can send an encrypted mail to a recipient, e.g. bobsmith@gmail.com, without the requiring either the existence of a Public-Key Infrastructure or that the recipient be on-line at the time of creation.

One common feature of all previous Identity-Based Encryption systems is that they view identities as a string of characters. In the paper the system propose a new type of Identity-Based Encryption that we call Fuzzy Identity-Based Encryption in which we view identities as a set of descriptive attributes. In a Fuzzy Identity-Based Encryption scheme, a user with the secret key for the identity  $I$  is able to decrypt a cipher text encrypted with the public key  $PK_I$  if and only if  $I$  and  $PK_I$  are within a certain distance of each other as judged by some metric. Therefore, the system allows for a certain amount of error-tolerance in the identities. Fuzzy-IBE gives rise to two interesting new applications. The first is an Identity-Based Encryption system that uses biometric identities. That is the system can view a user's biometric, for example an iris scan, as that user's identity described by several attributes and then encrypt to the user using their biometric identity. Since biometric measurements are noisy, cannot use existing IBE systems. However, the error-tolerance property of Fuzzy-IBE allows for a private key (derived from a measurement of a biometric) to decrypt a cipher text encrypted with a slightly different measurement of the same biometric.

## **2.12 Parallel and Dynamic Searchable Symmetric Encryption :**

Seny Kamara [6] (2013) et al proposed Searchable symmetric encryption (SSE) enables a client to outsource a collection of encrypted documents in the cloud and retain the ability to perform keyword searches without revealing information about the contents of the documents and queries. Although efficient SSE constructions are known, previous solutions are highly sequential. This is mainly due to the fact that, currently, the only method for achieving sub-linear time search is the inverted index approach (Curtmola, Garay, Kamara and Ostrovsky, CCS '06) which requires the

search algorithm to access a sequence of memory locations, each of which is unpredictable and stored at the previous location in the sequence.

Motivated by advances in multi-core architectures, The system present a new method for constructing sub-linear SSE schemes. The approach is highly parallelizable and dynamic. With roughly a logarithmic number of cores in place, searches for a keyword  $w$  in our scheme execute in  $o(r)$  parallel time, where  $r$  is the number of documents containing keyword  $w$  (with more cores, this bound can go down to  $O(\log n)$ , i.e., independent of the result size  $r$ ). Such time complexity outperforms the optimal( $r$ ) sequential search time—a similar bound holds for the updates. Our scheme also achieves the following important properties: (a) it enjoys a strong notion of security, namely security against adaptive chosen-keyword attacks; (b) compared to existing sub-linear dynamic SSE schemes (e.g., Kamara, Papamanthou, Roeder, CCS '12), updates in our scheme do not leak any information, apart from information that can be inferred from previous search tokens; (c) it can be implemented efficiently in external memory (with logarithmic I/O overhead).

The technique is simple and uses a red-black tree data structure; its security is proven in the random oracle model. Cloud storage promises high data availability, easy access to data, and reduced infrastructure costs by storing data with remote third-party providers. But availability is often not enough, as clients need privacy guarantees for many kinds of sensitive data that is outsourced to untrusted providers. For example, privacy is clearly important for medical data, enterprise data and secret government documents. The standard approach to achieving privacy in storage systems is to encrypt data using symmetric encryption.

Storage systems based on this approach provide end-to-end privacy in the sense that data is protected as soon as it leaves the client's possession. While such a solution provides strong security guarantees, it induces a high cost in terms of functionality and is therefore inadequate for storage systems that handle data at large scales. This is because after the data leaves the client's machine in encrypted form, the server cannot perform any meaningful computation on it. To address this, one can either use general-purpose solutions (e.g., fully-homomorphic encryption or oblivious RAMs ) or special-purpose solutions (e.g., searchable encryption). The approach is highly parallelizable and dynamic. Cloud storage promises high data availability, easy access to data, and reduced infrastructure costs by storing data with remote third-party providers .Although general-purpose

solutions have advantages, including generality and stronger security properties, they are mostly of theoretical interest (e.g., recent work has shown that ORAM can be relatively practical). On the other hand, special-purpose solutions like searchable encryption are practical and aim to provide a reasonable trade-off between efficiency, functionality and security.

### **2.13 Trusted Launch of Virtual Machine Instances in Public IaaS Environments :**

Nicolae Paladi [8] (2013), Cloud computing and Infrastructure-as-a-Service (IaaS) are emerging and promising technologies, however their adoption is hampered by data security concerns. At the same time, Trusted Computing (TC) is experiencing an increasing interest as a security mechanism for IaaS. In this paper we present a protocol to ensure the launch of a virtual machine (VM) instance on a trusted remote compute host. Relying on Trusted Platform Module operations such as binding and sealing to provide integrity guarantees for clients that require a trusted VM launch, the system have designed a trusted launch protocol for VM instances in public IaaS environments. The system also present a proof-of-concept implementation of the protocol based on OpenStack, an open-source IaaS platform.

The results provide a basis for the use of TC mechanisms within IaaS platforms and pave the way for a wider applicability of TC to IaaS security. One of the distinguished trends in IT operations today is the consolidation of IT systems onto common platforms. A key technology in realizing this is system virtualization. System virtualization makes it possible to streamline IT operations, save energy and obtain better utilization of hardware resources. A virtualized computing infrastructure allows clients to run own services in form of Virtual Machines (VM) on shared computing resources.

The approach however introduces new challenges, as it means that information previously controlled by one administrative domain and organization, is now under the control of a third party provider and that the information owner loses direct control over how data and services are used and protected. IaaS is one of the business models based on system virtualization and security aspects are among the main identified obstacles for its adoption [3]. The problems with securing IaaS are evident not least through the fact that widely known platforms such as Amazon EC2, Microsoft

Azure, services provided by Rack Space and other IaaS services are plagued by vulnerabilities at several levels of the software stack, from the web based cloud management console to VM side-channel. Relying on Trusted Platform Module operations such as binding and sealing.

## **2.14 Domain-Based Storage Protection (DBSP) in Public Infrastructure Clouds :**

Nicolae Paladi [10] (2013) et al proposed , Confidentiality and integrity of data in Infrastructure-as-a-Service (IaaS) environments increase in relevance as adoption of IaaS advances towards maturity. While current solutions assume a high degree of trust in IaaS provider and infrastructure management processes, earlier incidents have demonstrated that neither are impeccable. In the paper the system introduces Domain-Based Storage Protection (DBSP) a data confidentiality and integrity protection mechanism for IaaS environments, which relies on trusted computing principles to provide transparent storage isolation between IaaS clients. The system describes the building blocks of this mechanism and provides a set of detailed protocols for generation and handling of keys for confidentiality and integrity protection of data stored by guest VM instances.

The protocols assume an untrusted IaaS provider and aim to prevent both malicious and accidental faulty configurations that could lead to breach of data confidentiality and integrity in IaaS deployments. Following a period of establishment and early adoption, cloud computing is gaining widespread popularity and is now present in the product portfolio of many large software vendors in one of the three archetypes outlined by the US National Institute of Standards and Technology (NIST): Infrastructure-as-a-Service, Platform-as-a-Service or Software-as-a-Service. Other factors which testify to the impact of the field are the emergence of legal frameworks that regulate provisioning and usage of public cloud computing services and protection of data transferred to public cloud storage. However, despite growing popularity, cloud computing continues to present a wide range of unsolved security concerns. Considering that security concerns were long cited as barriers to wider adoption of public cloud services, emerging regulation will likely require public cloud providers to operate with an even wider set of tools to safeguard when needed the confidentiality, integrity, authenticity and even geolocation of data stored in public

clouds. Governmental programs, such as e.g FedRAMP in the USA propose a "standardized approach to security assessment, authorization, and continuous monitoring for cloud products and services".

Programs are an important extension of the cloud security ecosystem, they often assume manual execution steps which cannot reliably exclude audit or reporting errors due to human factors. In addition, the outcome of such programs is a certification result based on a snapshot view of the public cloud providers' infrastructure, processes and policies, while an adversary with full logical access to the underlying infrastructure can conceal traces of an eventual security breach. Thus, while security assessment and continuous monitoring of the infrastructure are valuable tools in ensuring security of IaaS infrastructure deployments, the system consider that effective prevention mechanisms have a higher potential to increase the IaaS customers' trust with respect to data processing and storage in public IaaS environments.

## **2.15 On the Security of Public Key Protocols :**

D.Dolcv [3] (1983) et al proposed the use of public key encryption to provide secure network communication has received considerable attention. Such public key systems are usually effective against passive eavesdroppers, who merely tap the lines and try to decipher the message. It has been pointed out, however, that an improperly designed protocol could be vulnerable to an active saboteur, one who may impersonate another user or alter the message being transmitted. In the paper the system formulate several models in which the security of protocols can be discussed precisely.

Algorithms and characterizations that can be used to determine protocol security in these models will be given. The use of public key encryption (Diffie and Hellman, Rivest, Shamir, and Adleman) to provide secure network communication has received considerable attention (Diffie and Hellman, Merkle , Needham and Schroeder, Popek and Kline). Such public key systems are usually very effective against a “passive” eavesdropper, namely, one who merely taps the communication line and tries to decipher the intercepted message. However, as pointed out in Needham and Schroeder, an improperly designed protocol could be vulnerable to an “active” saboteur, one who may impersonate another user and may alter or replay the message. As a protocol might be compromised in a complex way, informal arguments that assert the security for a protocol are prone to errors. It



is thus desirable to have a formal model, in which the security issues can be discussed in a precise manner. The models the system introduce later will enable us to study the security problem for families of protocols, with very few assumptions on the behavior of the saboteur.

## **2.16 Terra: A Virtual Machine-Based Platform for Trusted Computing :**

Tal Garfinkel [5] (2003) et al proposed the paper present a flexible architecture for trusted computing, called Terra, that allows applications with a wide range of security requirements to run simultaneously on commodity hardware. Applications on Terra enjoy the semantics of running on a separate, dedicated, tamper-resistant hardware platform, while retaining the ability to run side-by-side with normal applications on a general purpose computing platform. Terra achieves this synthesis by use of a *trusted virtual machine monitor* (TVMM) that partitions a tamper-resistant hardware platform into multiple, isolated virtual machines (VM), providing the appearance of multiple boxes on a single, general-purpose platform. To each VM, the TVMM provides the semantics of either an “open box,” i.e. a general-purpose hardware platform like today’s PCs and work stations, or a “closed box,” an opaque special-purpose platform that protects the privacy and integrity of its contents like today’s game consoles and cellular phones.

The software stack in each VM can be tailored from the hardware interface up to meet the security requirements of its application(s). The hardware and TVMM can act as a trusted party to allow closed-box VMs to cryptographically identify the software they run, i.e. what is in the box, to remote parties. The system explore the strengths and limitations of this architecture by describing our prototype implementation and several applications that we developed for it. Commodity computing systems have reached an impasse. There is an increasing need to deploy systems with diverse security requirements in enterprise, government, and consumer applications. However, current hardware and operating systems impose fundamental limitations on the security these platforms can provide. First, commodity operating systems are complex programs that often contain millions of lines of code, thus they inherently offer low assurance. Building simple, high-assurance applications on top of these operating systems is impossible because applications ultimately depend on the operating system as part of their trusted computing base.

Next, commodity operating systems poorly isolate applications from one another. As a result, the compromise of almost any application on a platform often compromises the entire platform. Applications with diverse security requirements cannot be run concurrently, because the platform's security level is reduced to that of its most vulnerable application. Further, current platforms provide only weak mechanisms for applications to authenticate themselves to their peers. There is no complete and ubiquitous mechanism for distributed applications to verify the identities of programs they interact with.

For example, an online game server cannot tell whether it is interacting with a game client that will play fairly or one which has been subjected to tampering that will allow users to cheat. Finally, current platforms provide no way to establish a trusted path between users and applications. For example, an application for trading on financial markets has no way of establishing if its inputs are coming from a human user or a malicious program. Conversely, human users have no way of establishing whether they are interacting with a trusted financial application or with a malicious program impersonating that application.

## **2.17 SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes :**

Arvind Seshadri [16] (2007) et al proposed the paper propose SecVisor, a tiny hypervisor that ensures code integrity for commodity OS kernels. In particular, SecVisor ensures that only user-approved code can execute in kernel mode over the entire system lifetime. This protects the kernel against code injection attacks, such as kernel rootkits. SecVisor can achieve this property even against an attacker who controls everything but the CPU, the memory controller, and system memory chips. Further, SecVisor can even defend against attackers with knowledge of zero-day kernel exploits. The goal is to make SecVisor amenable to formal verification and manual audit, thereby making it possible to rule out known classes of vulnerabilities. To this end, SecVisor offers small code size and small external interface. The system rely on memory virtualization to build SecVisor and implement two versions, one using software memory virtualization and the other using CPU-supported memory virtualization.

The code sizes of the runtime portions of these versions are 1739 and 1112 lines, respectively. The size of the external interface for both versions of SecVisor is 2 hypercalls. It is

easy to port OS kernels to SecVisor. We port the Linux kernel version 2.6.20 by adding 12 lines and deleting 81 lines, out of a total of approximately 4.3 million lines of code in the kernel. Computing platforms are steadily increasing in complexity, incorporating an ever-growing range of hardware and supporting an ever-growing range of applications. Consequently, the complexity of OS kernels is steadily increasing. The increased complexity of OS kernels also increases the number of security vulnerabilities. The effect of these vulnerabilities is compounded by the fact that, despite many efforts to make kernels modular, most kernels in common use today are monolithic in their design.

A compromise of any part of a monolithic kernel could compromise the entire kernel. Since the kernel occupies a privileged position in the software stack of a computer system, compromising it gives the attacker complete control of the system. In view of the importance of the security of the kernel to the security of a system, securing existing kernels is of critical importance. Approaches that do not mandate large-scale design changes to existing kernels are preferable, since they will ease deployment. SecVisor represents a first step in that direction, as it provides a lifetime guarantee of the integrity of the code executing with kernel privilege. In other words, SecVisor prevents an attacker from either modifying existing code in a kernel or from executing injected code with kernel privilege, over the lifetime of the system. SecVisor can achieve this guarantee even in the presence of an attacker who controls everything on the system except for the CPU, memory controller, and system memory chip. SecVisor ensures that only code approved by the user can execute with kernel privilege. Users can supply their desired approval policy and SecVisor checks all code loaded into the kernel against the users' approval policy. It further ensures that the approved code currently in memory cannot be modified by the attacker.

## **2.18 Cloud Visor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization :**

Fengzhe Zhang [18] (2011) et al proposed Multi-tenant cloud, which usually leases resources in the form of virtual machines, has been commercially available for years. Unfortunately, with the adoption of commodity virtualized infrastructures, software stacks in typical multi-tenant clouds are non-trivially large and complex, and thus are prone to compromise or abuse from adversaries including the cloud operators, which may lead to leakage of security-sensitive data. In

the paper, the system propose a transparent, backward-compatible approach that protects the privacy and integrity of customers' virtual machines on commodity virtualized infrastructures, even facing a total compromise of the virtual machine monitor (VMM) and the management VM. The key of our approach is the separation of the resource management from security protection in the virtualization layer.

A tiny security monitor is introduced underneath the commodity VMM using nested virtualization and provides protection to the hosted VMs. As a result, our approach allows virtualization software (e.g., VMM, management VM and tools) to handle complex tasks of managing leased VMs for the cloud, without breaking security of users' data inside the VMs. Multi-tenant cloud has advantages of providing elastic and scalable computing resources and freeing users from the cumbersome tasks such as configuring, managing and maintaining IT resources.

For example, Amazon's Elastic Compute Cloud (EC2) platform provides flexible and resizable computing resources in the form of Xen-based VMs for a number of usage scenarios, including application hosting, content delivering, e-commerce and web hosting. However, multi-tenant cloud also redefines the threat model of computing and raises new security challenges: the security of customers' sensitive data will be a key concern if being put into a third party multi-tenant cloud. Unfortunately, current multi-tenant cloud platforms adopting commodity virtualization infrastructures usually provide limited assurance for the security of tenants' sensitive data. Many cloud providers only provide "security on your own" guarantee to users' content.

## **2.19 An Efficient Cryptographic Protocol Verifier Based on Prolog Rules :**

Bruno Blanchet [2] (2001) et al proposed a new automatic cryptographic protocol verifier based on a simple representation of the protocol by Prolog rules, and on a new efficient algorithm that determines whether a fact can be proved from these rules or not. This verifier proves secrecy properties of the protocols. Thanks to its use of unification, it avoids the problem of the state space explosion. Another advantage is that we do not need to limit the number of runs of the protocol to analyze it. The system have proved the correctness of our algorithm, and have implemented it. The experimental results show that many examples of protocols of the literature, including Scheme [24], can be analyzed by our tool with very small resources: the analysis takes from less than 0.1 s for

simple protocols to 23 s for the main mode of Scheme. It uses less than 2 Mb of memory in our tests. The design of cryptographic protocols is difficult and error-prone.

Most existing protocol verifiers based on model checking suffer from the problem of the state space explosion, and they need very large resources to verify even relatively simple protocols. Moreover, in general, they limit the number of runs of this work was partly done while the author was at Bell Labs Research, Lucent Technologies. Protocol to guarantee the termination of the verification process. If there exists an attack that only appears with more runs of the protocol, it will not be discovered. Our solution to these problems relies on two ideas: a simple intermediate representation of the protocols; a new efficient solving algorithm. We use Prolog rules to represent the protocol and the attacker. Messages and channels are represented by terms; the fact *attacker* means that the attacker has the message *M*; rules give implications between such facts. This can be considered as an abstraction of the multiset rewriting or of the linear logic representation. We perform two interesting abstractions: Fresh values considered as functions other messages in the protocol.

To give an intuition, when the attacker does not modify messages, different values are used for each pair of participants of the protocol, instead of per session. The system forget the number of times a message appear to remember only that it has appeared. A step of the protocol can be executed several times instead of only once in each session. These are keys to avoid limiting the number of runs of protocols: the number of repetitions is simply forgotten. Our approximations are safe, in the sense that if the verifier does not find a flaw in the protocol, then there is no flaw. The verifier therefore provides real security guarantees. In contrast, it may give a false attack against the protocol. However, false attacks are rare, and we have been able to prove the secrecy properties of all the protocols that we have considered.

## **CHAPTER 3**

### **SYSTEM ANALYSIS**

#### **3.1 EXISTING SYSTEM :**

In Existing system, a Data Sharing system model, there are multiple user security who may encrypt according to their own ways, possibly using different sets of cryptographic keys. Letting each user obtain keys from every owner who's their central idea talks about the impossibility of Fully Homomorphic Encryption (FHE) alone for VM Cloud privacy. Their classification hierarchy of VM Cloud Computing is not standard model and has few shortcomings as we would discuss duly. The system states the security and privacy issues from a standard VM Cloud Computing definitions and discuss the challenges involved not just for FHE but also for many other techniques, but this requires too much trust on a single authority (i.e., cause the key escrow problem).

Elliptical Curve Cryptography is an arrangement in which the keys needed to decrypt encrypted data are held in ECC so that, under certain circumstances, an authorized third party may gain access to those keys. These third parties may include businesses, who may want access to employees' private communications, or governments, who may wish to be able to view the contents of encrypted communications.

##### **3.1.2 Drawbacks of Existing System :**

- Key information depends on centralized key server.
- Computational and Communication cost is more.
- More resources used for rekeying because it is being done for individual join/leave operation.
- High in Memory usage and encryption key length.
- Data Transmission time and execution is high.

## 3.2 PROPOSED SYSTEM :

Proposed System endeavor to study the patient centric, solves the problem of evaluating a function jointly by multiple parties on their private inputs secure sharing of file sharing in VM Cloud stored on semi-trusted servers, and focus on addressing the complicated and challenging key management issues. It also no assumptions are made on computational resources available with the parties. All the parties would carry out same amount of work which is contrary to VM Cloud Computing setting.

To adapt these techniques for an asymmetric setting like VM Cloud Computing where the server has massive amounts of computing power relative to the users, In order to protect the personal health data stored on a semi-trusted server, we adopt Diffie Hellman is better than ECC as the main encryption primitive.

Precise lower bounds on hard computations, but complexity theorists have had limited success in establishing lower bounds in general, so instead we reason relatively: we show that the hard computations are at least as hard as solving some problem known or assumed (usually the latter, for reasons to be explained in due course) to be hard.

The proof technique for making assertions about the complexity of one problem on the basis of another is called reduction “Using DH, access policies are expressed based on the attributes of users or data, which enables a patient to selectively share her file sharing among a set of users by encrypting the file under a set of attributes, without the need to know a complete list of users. The complexities per encryption, key generation and decryption are only linear with the number of attributes involved.

### 3.2.1 Advantages of Proposed System :

- Key information does should be depend on VM Cloud centralized key server.
- Computational and Communication cost is less.
- Resources used for rekeying is minimized because it is being done for batch of join/leave operations.

- More secure by Boolean logic minimization because session management done by this concept.
- Low Memory Usage.
- High Throughput.



## CHAPTER 4

### SYSTEM DESIGN

#### 4.1 DATA FLOW DIAGRAM :

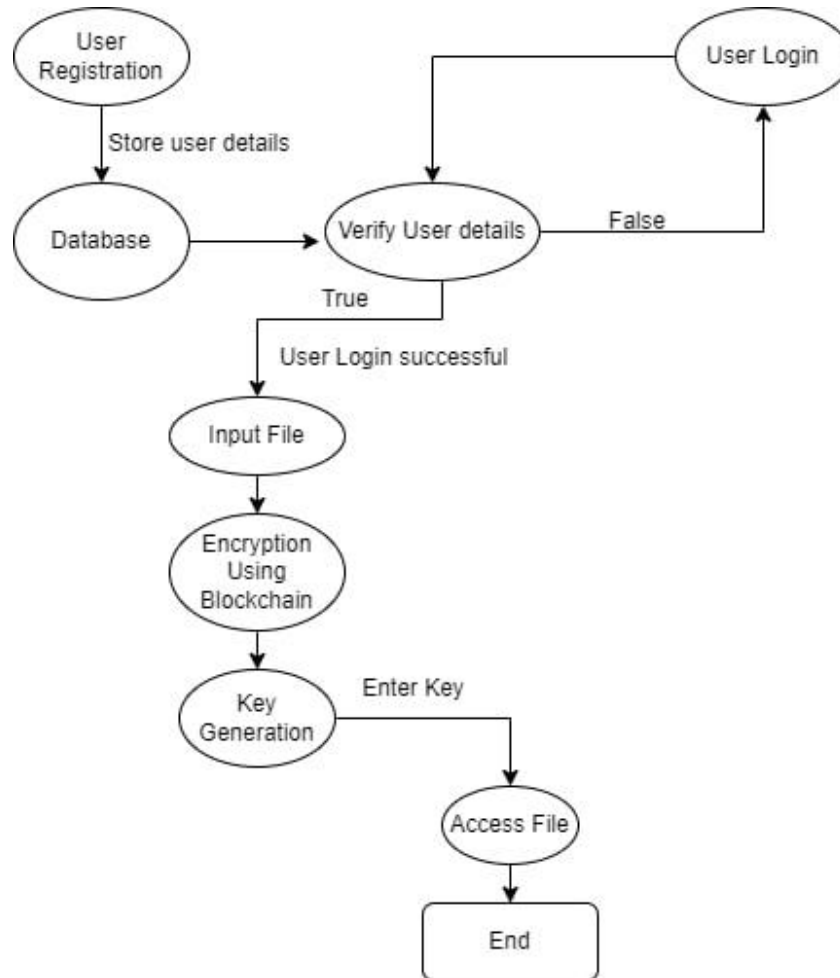


Figure 4.1 Data Flow Diagram

## 4.2 ALGORITHM ARCHITECTURE :

### 4.2.1 AES ALGORITHM :

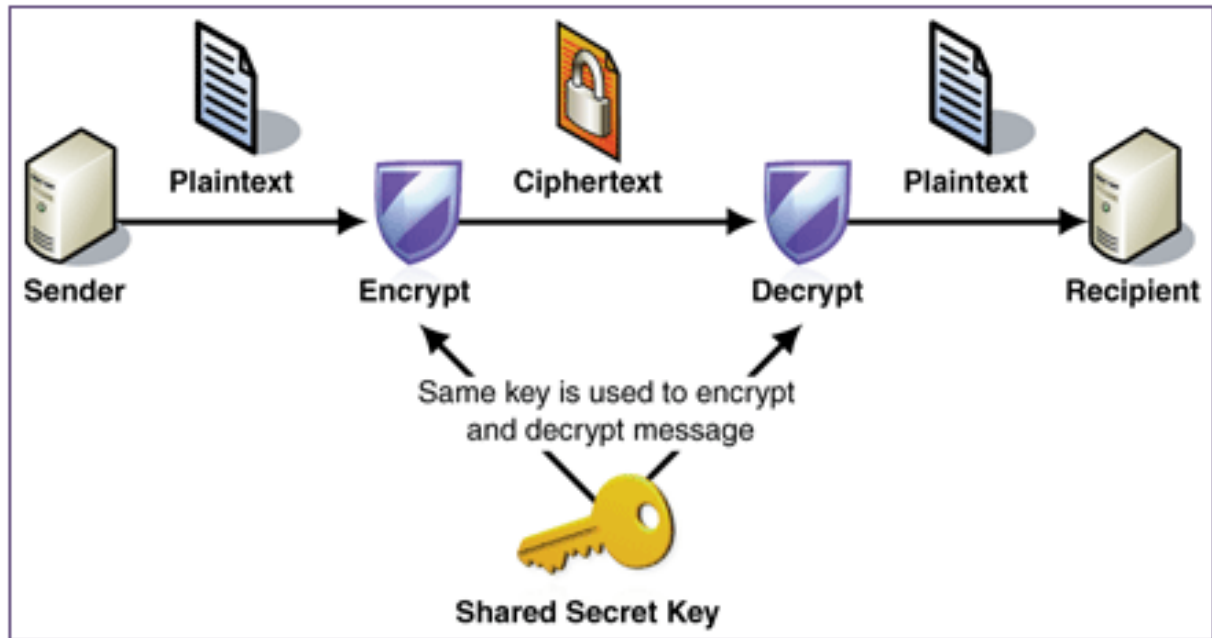


Figure 4.2 AES Architecture

### 4.2.2 RSA ALGORITHM :

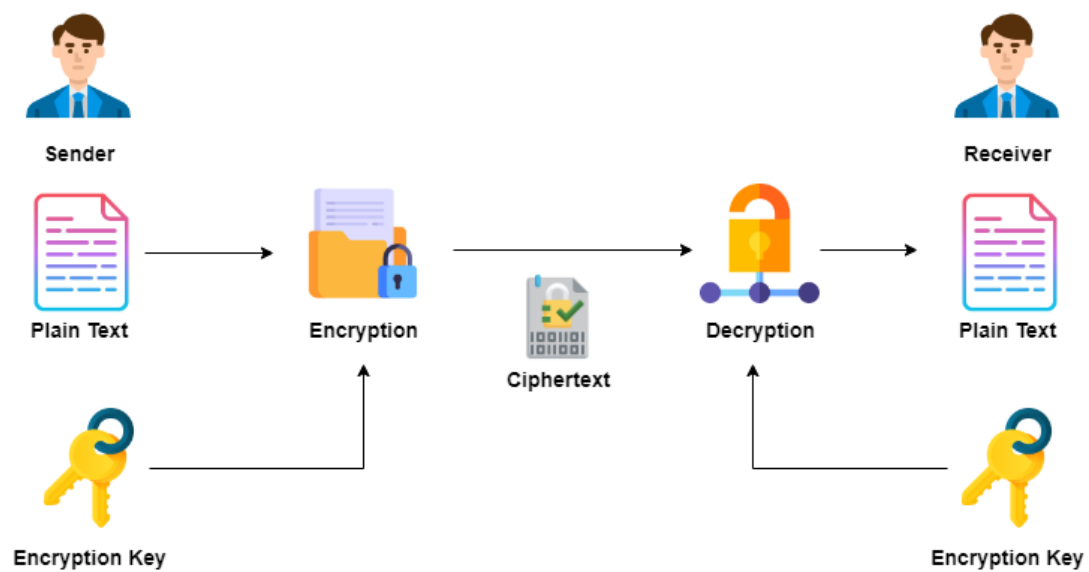


Figure 4.3 RSA Architecture

### 4.2.3 DH ALGORITHM :

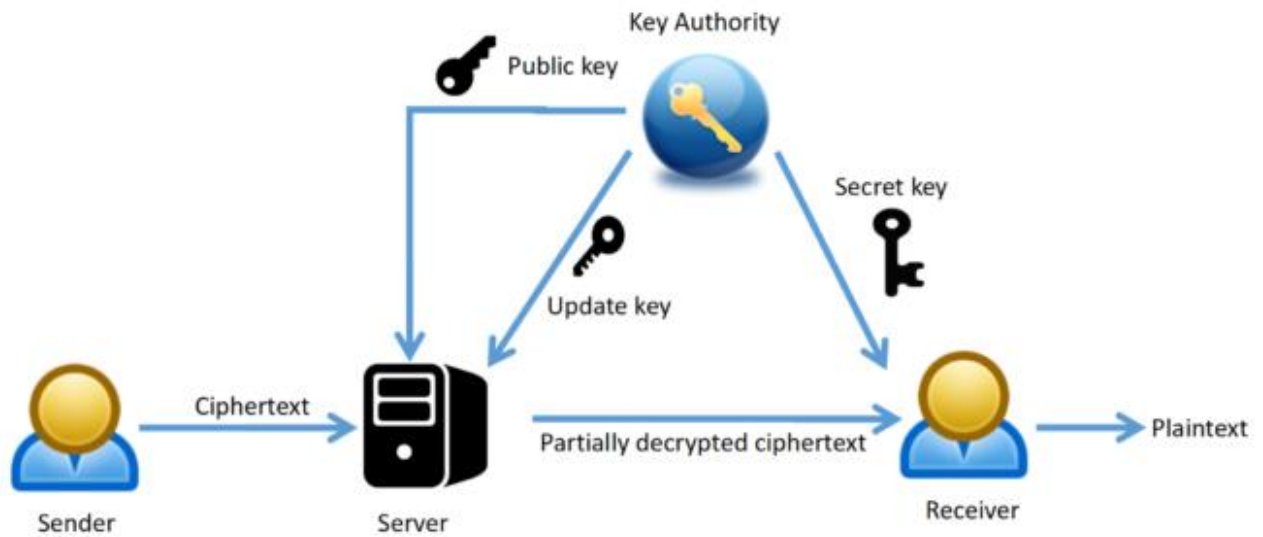


Figure 4.4 DH Architecture

### 4.2.4 ECC ALGORITHM :

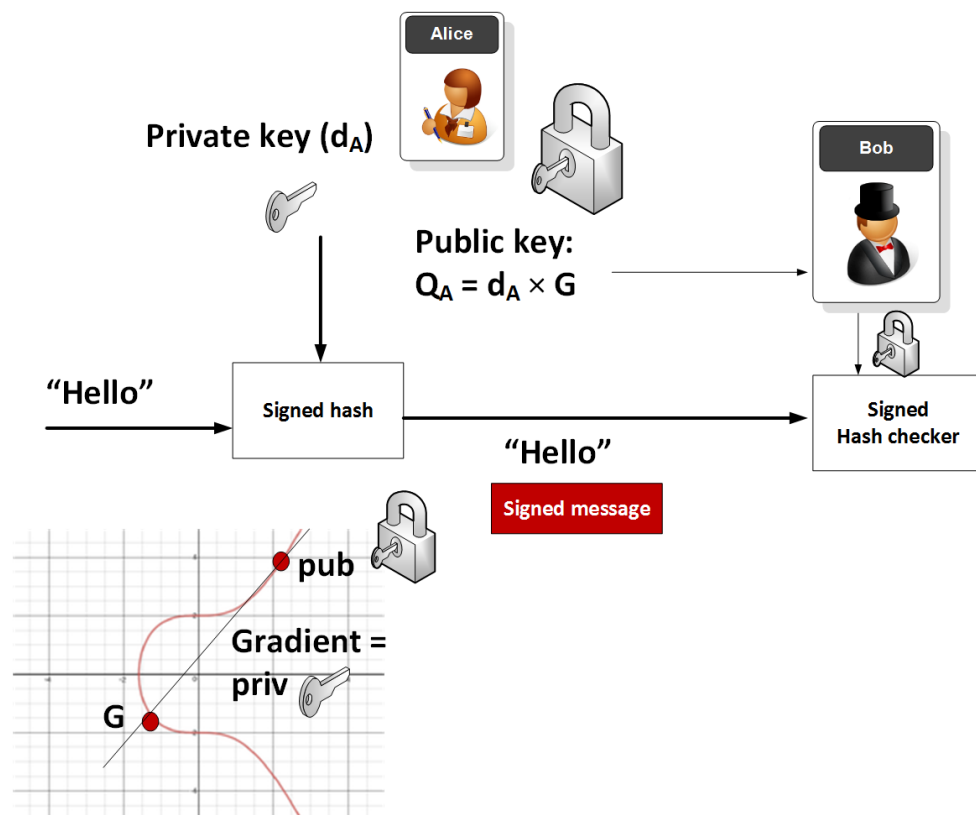


Figure 4.5 ECC Architecture

## CHAPTER 5

### ALGORITHMS USED IN BLOCKCHAIN

#### 5.1 DIFFIE-HELLMAN ALGORITHM :

Diffie-Hellman key exchange offers the best of both worlds -- it uses public key techniques to allow the exchange of a private encryption key. Let's take a look at how the protocol works, from the perspective of Alice and Bob, two users who wish to establish secure communications. We can assume that Alice and Bob know nothing about each other but are in contact.

Here are the nine steps of the process:

- 1) Communicating in the clear, Alice and Bob agree on two large positive integers,  $n$  and  $g$ , with the stipulation that  $n$  is a prime number and  $g$  is a generator of  $fn$ .
- 2) Alice randomly chooses another large positive integer,  $X_A$ , which is smaller than  $n$ .  $X_A$  will serve as Alice's private key.
- 3) Bob similarly chooses his own private key,  $X_B$ .
- 4) Alice computes her public key,  $Y_A$ , using the formula  $Y_A = (g^{X_A}) \bmod n$ .
- 5) Bob similarly computes his public key,  $Y_B$ , using the formula  $Y_B = (g^{X_B}) \bmod n$ .
- 6) Alice and Bob exchange public keys over the insecure circuit.
- 7) Alice computes the shared secret key,  $k$ , using the formula  $k = (Y_B^{X_A}) \bmod n$ .
- 8) Bob computes the same shared secret key,  $k$ , using the formula  $k = (Y_A^{X_B}) \bmod n$ .
- 9) Alice and Bob communicate using the symmetric algorithm of their choice and the shared secret key,  $k$ , which was never transmitted over the insecure circuit.

#### 5.2 RSA ALGORITHM :

RSA (Rivest–Shamir–Adleman) is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called public key cryptography, because one of the keys can be given to anyone. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes that the Public Key is given to everyone and Private key is kept

private. RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024 bit keys could be broken in the near future. But till now it seems to be an infeasible task.

### 5.3 AES ALGORITHM :

The Advanced Encryption Standard (AES) is a symmetric block cipher to protect classified information. AES is implemented in software and hardware throughout the world to encrypt sensitive data. It is essential for government computer security, cybersecurity and electronic data protection.

It uses 128,192, or 256 bit keys to transform a block of 128 bits message into a 128 bits of cipher text which is the main reason why it is strong, secure and exponentially stronger than the DES that uses 56 bit key. A substitution-permutation, or SP network, with several rounds is used by the AES algorithm to generate cipher text. The key length used will determine the number of rounds. For example, in the encryption process the 128 bit keys consist of 10 rounds, 12 rounds for the 192 bit keys and 14 rounds for 256 bit keys. In each case, all the rounds are identical except for the last round we won't have the mix column step. Instead of having a one line of bytes or bits like most ciphers, AES arranges them in a 4x4 grid.

Byte 00	Byte 04	Byte 08	Byte 12
Byte 01	Byte 05	Byte 09	Byte 13
Byte 02	Byte 06	Byte 10	Byte 14
Byte 03	Byte 07	Byte 11	Byte 15

## 5.4 ECC ALGORITHM :

ECC, an alternative technique to RSA, is a powerful cryptography approach. It generates security between key pairs for public key encryption by using the mathematics of elliptic curves.

RSA does something similar with prime numbers instead of elliptic curves, but ECC has gradually been growing in popularity recently due to its smaller key size and ability to maintain security. This trend will probably continue as the demand on devices to remain secure increases due to the size of keys growing, drawing on scarce mobile resources. This is why it is so important to understand elliptic curve cryptography in context.

In contrast to RSA, ECC bases its approach to public key cryptographic systems on how elliptic curves are structured algebraically over finite fields. Therefore, ECC creates keys that are more difficult, mathematically, to crack. For this reason, ECC is considered to be the next generation implementation of public key cryptography and more secure than RSA.

It also makes sense to adopt ECC to maintain high levels of both performance and security. That's because ECC is increasingly in wider use as websites strive for greater online security in customer data and greater mobile optimization, simultaneously. More sites using ECC to secure data means a greater need for this kind of quick guide to elliptic curve cryptography.

An elliptic curve for current ECC purposes is a plane curve over a finite field which is made up of the points satisfying the equation :  $y^2 = x^3 + ax + b$ .

In this elliptic curve cryptography example, any point on the curve can be mirrored over the x-axis and the curve will stay the same. Any non-vertical line will intersect the curve in three places or fewer.

## CHAPTER 6

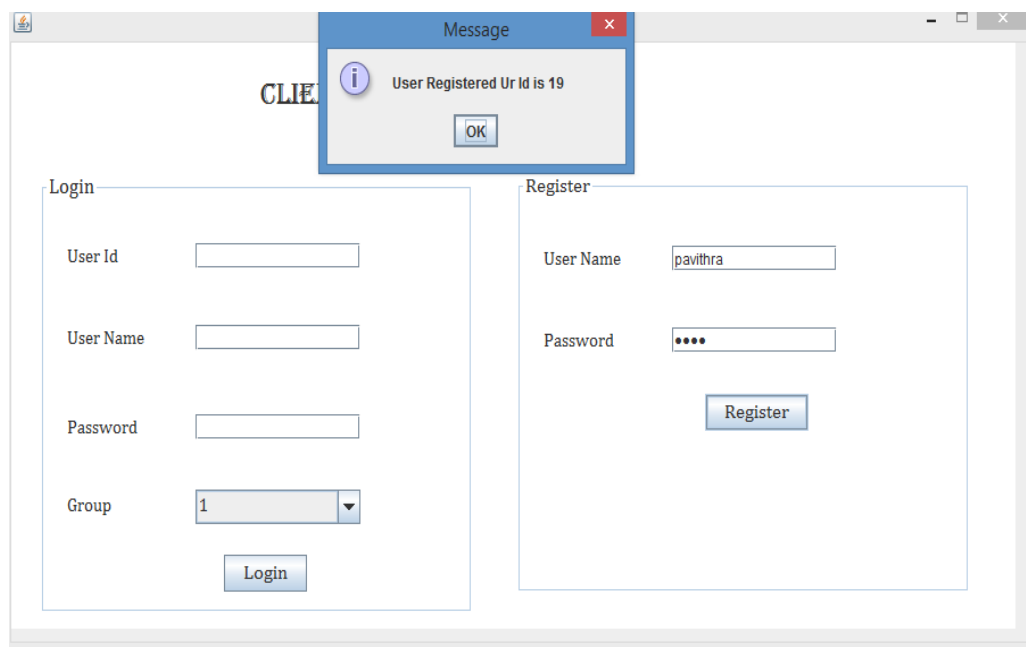
### SYSTEM MODEL

The efficient providing user security guarantees in public infrastructure cloud provides to mainly focus as following modules,

- Registration and Encryption
- Database Storage
- Group Key Generation within the workgroup
- Keying and Rekeying the group key
- Sharing the data within workgroup

#### 6.1 Registration and Encryption:

The client module the client program was implemented using Java servlets and a JFrame page that invokes the servlet. The user enters the data to be sent via the JFrame page which then invokes the Client servlet. The servlet then encrypts this data using the shared key object generated by the Diffie-Hellman Key Agreement algorithm and the Data Encryption Standard (in ENCRYPT mode) and send it over to the server. The client served uses URL Redirection to send the encrypted message from the client to the server.



**Figure 6.1 Registration**



The image shows a Windows application window titled "Client". The main content area is titled "CLIENT LOGIN REGISTER". It is divided into two sections: "Login" and "Register".

**Login Section:**

- User Id:
- User Name:
- Password:
- Group:  (with a dropdown arrow)
- 

**Register Section:**

- User Name:
- Password:
- 

**Figure 6.2 Login**

The image shows a Windows application window titled "Client-19". The main content area is titled "PUBLIC KEY ENCRYPTION - CLIENT". There is a "Leave" button in the top right corner.

Below the title bar, there is a tabbed interface with the following tabs: "Encrypt File", "Encrypted Text", "Enc Time", "Key Gen", "File Sharing", and "View File". The "Encrypt File" tab is currently selected.

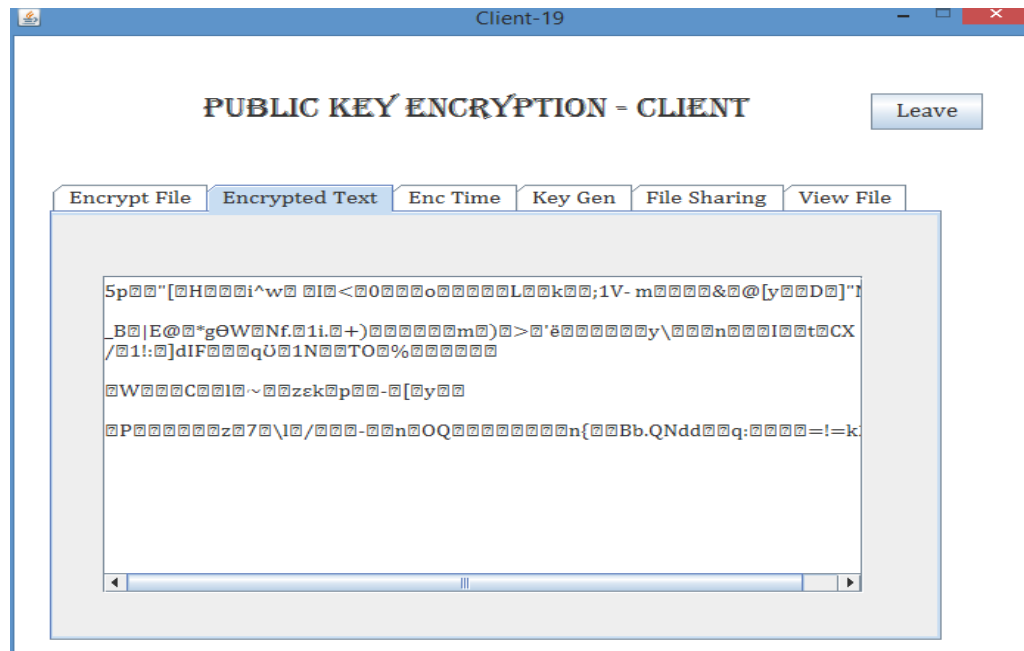
Inside the "Encrypt File" tab, there is a section titled "Choose Text File". It contains a text box with the file path "Users\Best\Desktop\New Text Document.txt" and a "Browse" button to its right.

Below the file path, there is a large text area containing the text: "hai welcome good morning have a nice day please smile".

At the bottom of the text area, there is an "Encrypt" button.

**Figure 6.3 Choosing File**

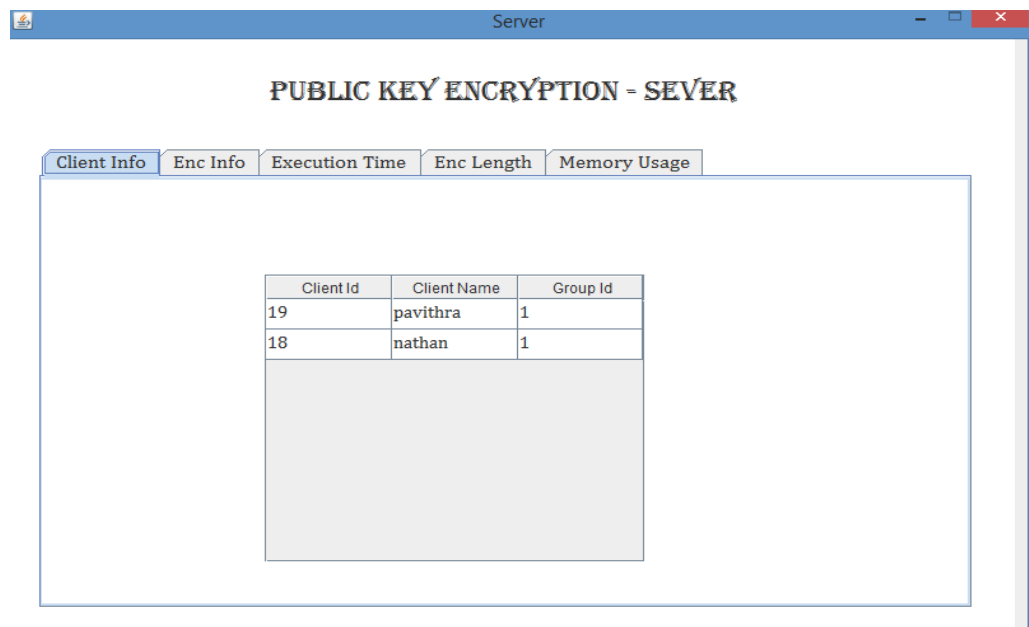




**Figure 6.4 Encrypted Text**

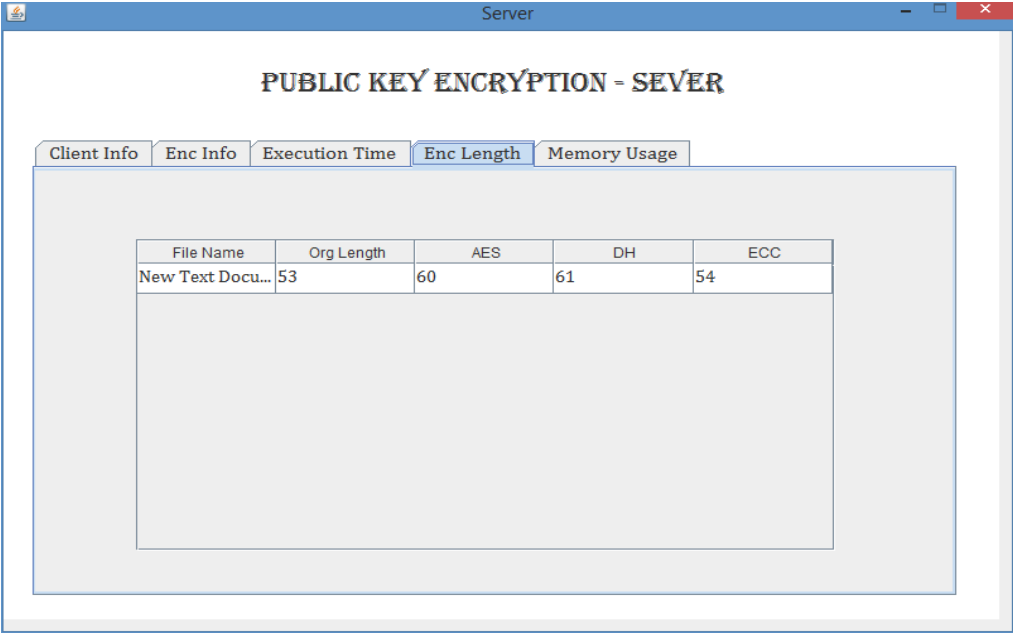
## 6.2 Database Storage :

The server itself is a simple servlet that is connected to a database. It receives the encrypted message from the client and decrypts it using the shared key object generated by the Diffie-Hellman algorithm and Diffie Hellman (in DECRYPT mode).



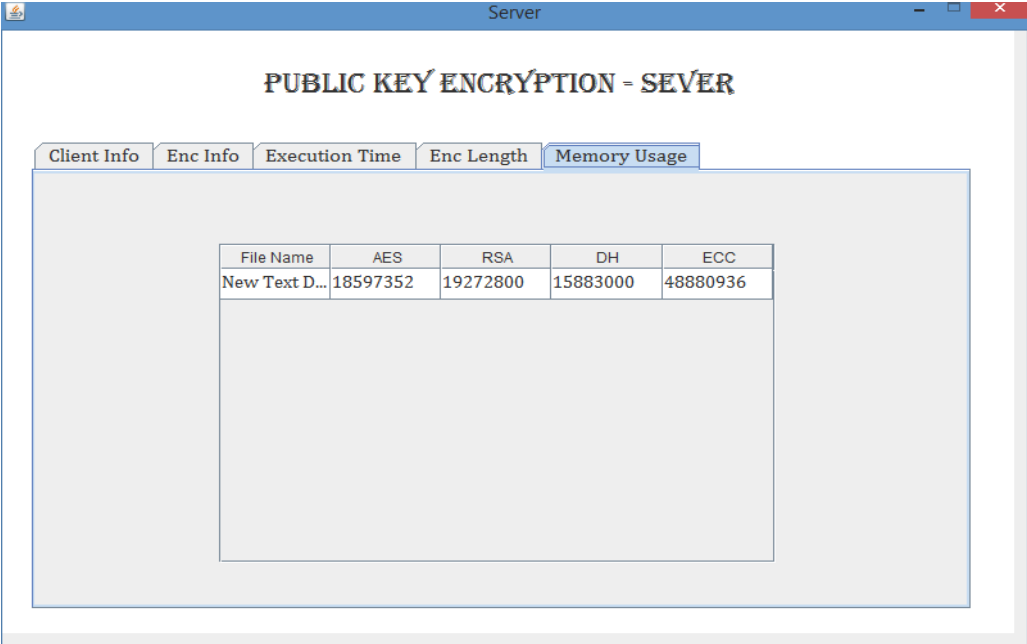
**Figure 6.5 Client Information**

Once the message has been encrypted the server will store the message into the database, which can be retrieved at a later stage.



File Name	Org Length	AES	DH	ECC
New Text Docu...	53	60	61	54

**Figure 6.6 Encryption Length**



File Name	AES	RSA	DH	ECC
New Text D...	18597352	19272800	15883000	48880936

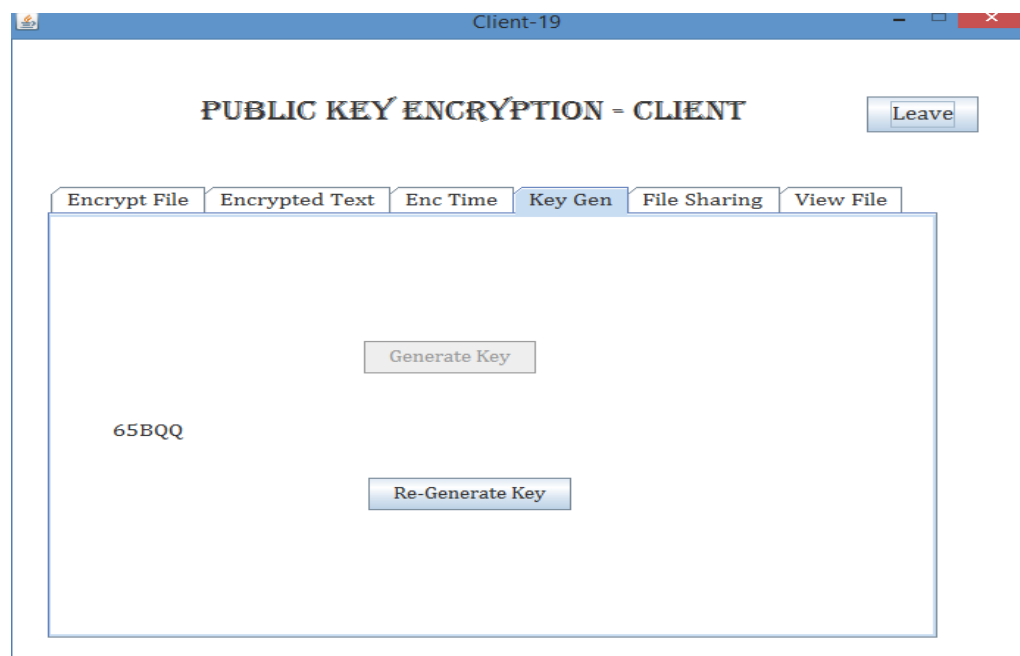
**Figure 6.7 Memory usage**

### 6.3 Group Key Generation within the workgroup :

The nodes in the workgroup will form a group key. Each group member will collaboratively contribute its part to the global group key. The group key is generated in a shared and contributory fashion and there is no single-point-of-failure. We are going to generate a group key. The group members are arranged in a logical key hierarchy known as a key tree. In the distributed key agreement protocols we consider, however, there is no centralized key server available. Moreover, an advantage of distributed protocols over the centralized protocols is the increase in system reliability, because the group key is generated in a shared and contributory fashion and there is no single-point-of-failure.

To efficiently maintain the group key in a dynamic peer group with more than two members, we use the tree-based group Elliptic Curve Diffie Hellman protocol. Each member maintains a set of keys, which are arranged in a hierarchical binary tree.

Each leaf node in the tree corresponds to the individual secret and blinded keys of a group member  $M_i$ . Therefore, the secret key held by the root node is shared by all the members and is regarded as the group key. Key tree used in the tree-based group Elliptic Curve Diffie Hellman protocol.



**Figure 6.8 Key Generation**

Rekeying the group key which means renewing the keys associated with the nodes of the key tree, this is performed whenever there is any group membership change including any batch of member joins the group. Rekeying means a new group key will be generated by members in the group. rekeying is also performed whenever there is any group membership change including any batch of existing members leaving the group. We find that the previous approaches perform all rekeying steps at the beginning of every rekeying interval. This results in high processing load during the update instance and thereby delays the start of the secure group communication. Thus, we propose a more effective algorithm which we call the Elliptic Curve Diffie Hellman algorithm. Its intuition is to reduce the rekeying load by pre-processing the joining members during the idle rekeying interval.

The Elliptic Curve Diffie Hellman algorithm is divided into two phases, namely the Queue-sub tree phase and the Queue-merge phase. The first phase occurs whenever a new member joins the communication group during the rekeying interval. In this case, we append this new member in a temporary key tree. The second phase occurs at the beginning of every rekeying interval and we merge the temporary tree (which contains all newly joining members) to the existing key tree.

#### 6.4 Sharing data within the workgroup :

The screenshot shows a web application window titled "Client". Inside, there's a header "CLIENT LOGIN REGISTER". Below it, there are two panels. The left panel, titled "Login", contains four input fields: "User Id" with the value "18", "User Name" with the value "nathan", "Password" with masked characters ".....", and "Group" with the value "1" and a dropdown arrow. A "Login" button is at the bottom of this panel. The right panel, titled "Register", contains two input fields: "User Name" and "Password", both empty. A "Register" button is at the bottom of this panel.

**Figure 6.9 Login another user in the same group**

With the help of group key generated by the members in the group, the data will be shared securely among the group. The group members will share the resources, namely accessing the files. We are implementing this with RMI (Remote Method Invocation). This feature aids in building distributed applications.



**Figure 6.10 File Sharing**

A *remote object* is one whose methods can be invoked from another Java virtual machine, potentially on a different host. An object of this type is described by one or more *remote interfaces* written in the Java programming language. A reference to a remote object can be passed as an argument or returned as a result in any method invocation.



**Figure 6.11 View File**

## **CHAPTER 7**

### **RESULTS AND DISCUSSIONS**

The main purpose of our study is to determine whether there is any gap between cryptographic protocol/scheme (in term of theoretical) and its engineering implementation. Our scheme will be integrated with the security factors with respect to the fact that solving the proposed method is very challenging, and that the shared key (i.e. the secret) is never itself transmitted over the channel.

Our Algorithm utilizes basic scientific ideas making execution simpler and in addition avoidance from common Attacks. Security change is useful in light of the fact that proposed Algorithm is the premise of a few security standards and services on the internet, and if the security. Diffie Hellman key trade approach for key distribution gives off an impression of being one of the favored systems utilized as a part of practice today.

## **CHAPTER 8**

### **CONCLUSION**

The Cloud computing as a technology would be adopted if the areas of concerns like security of the data will be covered with full proof mechanism. The strength of cloud computing is the ability to manage risks in particular to security issues. Our suggested model will present an outline sketch of architecture to be adopted by architects involved in implementing the cloud computing. Security algorithms mentioned for encryption and decryption and ways proposed to access the multimedia content can be implemented in future to enhance security framework over the network.

The proposed system explore our work by providing algorithm implementations and producing results to justify our concepts of security for cloud computing. In order for this approach to work as intended, the cloud service provider must co-operate with the user in implementing solution. Some cloud service providers base their business models on the sale of user data to advertisers. These providers probably would not be willing to allow the user to use their applications in a way that preserves user privacy.



# **APPENDICES**

## **APPENDIX 1**

### **SOFTWARE AND HARDWARE DESCRIPTION**

#### **SOFTWARE REQUIREMENTS :**

Wamp server

Eclipse – IDE

#### **HARDWARE REQUIREMENTS :**

Processor : Pentium i3

Speed : 3.40 GHZ

Hard disk : 500GB

Ram : 4 GB DD2 RAM

## APPENDIX 2

### **CODING :**

#### **Client :**

##### **Main.java**

```
package client;

import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class Main {

    public static void main(String[] args) {

        // TODO code application logic here

        LoginFrame lf=new LoginFrame();

        lf.setVisible(true);

        lf.setTitle("Client");

        lf.setResizable(false);

        /*String id=JOptionPane.showInputDialog(new JFrame(),"Enter Client Id");

        ClientFrame cf=new ClientFrame(id);

        cf.setVisible(true);

        cf.setTitle("Client - "+id);

        cf.setResizable(false);

        ClientReceiver cr=new ClientReceiver(cf,id);

        cr.start();*/

    }

}
```

**DBConnection.java**

```
package client;

import java.sql.Statement;
import java.sql.Connection;
import java.sql.DriverManager;

public class DBConnection
{
    public Statement stt;
    public Connection con;
    public DBConnection()
    {
        try
        {
            Class.forName("com.mysql.cj.jdbc.Driver");

            con=DriverManager.getConnection("jdbc:mysql://localhost:3306/groupfilessharing","root","");

            stt=con.createStatement();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

}

## REFERENCES

- [1] M. Aslam, C. Gehrman, L. Rasmusson, and M. Björkman,(2012), “Securely launching virtual machines on trustworthy platforms in a public cloud - an enterprise’s perspective.,” in CLOSER, pp. 511– 521, SciTePress,.
- [2] B. Blanchet,(2001), “An efficient cryptographic protocol verifier based on prolog rules,” in Computer Security Foundations Workshop, IEEE, pp. 0082–0082, IEEE Computer Society.
- [3] D. Dolev and A. C. Yao,(1983), “On the security of public key protocols,” Information Theory, IEEE Transactions on, vol. 29, no. 2.
- [4] S. Graf, P. Lang, S. A. Hohenadel, and M. Waldvogel,(2012) “Versatile key management for secure cloud storage,” Reliable Distributed Systems, IEEE Computer Society, pp. 469– 474.
- [5] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, (2003),“Terra: A virtual machine-based platform for trusted computing,” in ACM SIGOPS Operating Systems Review, vol. 37, ACM.
- [6] S. Kamara and C. Papamanthou,(2013), “Parallel and dynamic searchable symmetric encryption,” in Financial Cryptography and Data Security, pp. 258– 274, Springer.
- [7] Michalas, N. Paladi, and C. Gehrman,(2014), “Security aspects of e-health systems migration to the cloud,” in E-health Networking, Application & Services (Healthcom’ 14), pp. 228–232, IEEE.
- [8] N. Paladi, C. Gehrman, M. Aslam, and F. Morenius,(2013), “Trusted Launch of Virtual Machine Instances in Public IaaS Environments,” in Information Security and Cryptology (ICISC’12), vol. 7839 of Lecture Notes in Computer Science, pp. 309–323, Springer.
- [9] N. Paladi, C. Gehrman, and F. Morenius,(2013),“Domain-Based Storage Protection (DBSP) in Public Infrastructure Clouds,” in Secure IT Systems, pp. 279–296, Springer.

- [10] N. Paladi, A. Michalas, and C. Gehrman,(2014), “Domain based storage protection with secure access control for the cloud,” in Cloud Computing, ASIACCS ’14, (New York, NY, USA), ACM.
- [11] A.R. Sadeghi and C. St ¨ uble,( 2004) “Property-based attestation for computing platforms: Caring about properties, not mechanisms,” New Security Paradigms, NSPW ’04, (New York, NY, USA), pp. 67–77, ACM,.
- [12] Sahai(2007), “Ciphertext-policy attribute-based encryption,” on Security and Privacy, . A. Sahai and B. Waters,(2005), “Fuzzy identity-based encryption,” in Advances in Cryptology–EUROCRYPT, Springer,.
- [13] N. Santos, K. P. Gummadi, and R. Rodrigues,(2009), “Towards trusted cloud computing,”in Cloud Computing, HotCloud’09, (Berkeley, CA, USA), USENIX Association.
- [14] N. Santos, R. Rodrigues, K. P. Gummadi, and S. Saroiu,( 2012), “Policy- Sealed Data: A New Abstraction for Building Trusted Cloud Services,” in Presented as part of the 21st USENIX Security Symposium (USENIX Security 12), (Bellevue, WA), pp. 175–188, USENIX.
- [15] J. Schiffman, T. Moyer, H. Vijayakumar, T. Jaeger, and P. McDaniel,( 2010), “Seeding Clouds With Trust Anchors,” in Cloud Computing Security, CCSW ’10, (New York, NY, USA), pp. 43–46, ACM.
- [16] Seshadri, M. Luk, N. Qu, and A. Perrig,( 2007), “SecVisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity OSes,” ACM SIGOPS Operating Systems Review, vol. 41, no. 6.
- [17] W. Wang, Z. Li, R. Owens, and B. Bhargava,(2009), “Secure and efficient access to outsourced data,” in Cloud computing security, pp. 55–66, ACM.
- [18] F. Zhang, J. Chen, H. Chen, and B. Zang,(2011), “Cloudvisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization,” Operating Systems Principles, pp. 203–216, ACM.