JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 35, 697-718 (2019)
DOI: 10.6688/JISE.201905_35(3).0012

Auto Sec SDN-XTR: A Hybrid End to End Security Mechanism with Efficient Trace Representation on Open Stack Cloud

S. VATCHALA[1,+], S. RAVIMARAN[2] AND A. SATHISH[3]
[1,3]*Centre for Research*
Anna University
Chennai, 600025 India
E-mail: vatchalacse@gmail.com[1,+]*; sathishroever05@gmail.com*[3]
[2]*M.A.M. College of Engineering*
Trichy, 621105 India
E-mail: principalmamce@mamce.org[2]

Even though open stack boosts business agility, availability, and efficiency by providing a platform with on-demand, resource pooling, self-service, highly elastic, and measured services capabilities, it needs improvisation in the following blocks of open stack such as Neutron which provides the networking capability for Open Stack and Cinder Block acts as a storage component; due to centralized administration in the open stack cloud environment. The problem with these components is that more susceptible to external attacks, unpredictable responses depending on the network load and lack of maintaining the integrity since cinder block shares simultaneous access to the same data. In order to meet the above requirements this work has proposed an AutoSec SDN-XTR (Automated end to end Security in Software Defined Networks – Efficient and Compact Subgroup Trace Representation). In order to rectify the security challenges additionally an efficient security algorithm XTR is proposed for providing the encryption of the file content that also involves a trace operation to incorporate integrity checking. This provides efficient security by involving the Diffie-Hellman for key agreement (both the public key and private key) and ElGamal approach for encryption. Then after the networking process storage of the files content occur in the cinder block store environment. In the cinder store erasure codes algorithm is utilized for data recovery where less storage will be achieved since replicas are not utilized and duplication of file content will not be done instead only parity data will be created as in the concept of RAID (Redundant Array of Independent Disk). Now the unique data which are recovered in cinder block are already been secured by XTR encryption and should be effectively distributed.

Keywords: key agreement, encryption, file transmission, file recovery, auto sec SDN

1. INTRODUCTION

Open Stack is a cloud platform which manages the infrastructure and also consists of two types of nodes such as the control nodes that manage the resources of the customers and cloud providers and the compute nodes that run the virtual machine (VM) of the customers [1]. The open stack controls the large services like compute, storage, and network Resources through data center, It includes many components services like (1) compute service (nova); (2) image service (glance); (3) dashboard service (horizon); (4) block storage service (cinder); (5) metric service (telemetry); (6) identity service (key-

Received July 20, 2018; revised September 12, 2018; accepted January 16, 2019.
Communicated by Ren-Hung Hwang.
[+] Corresponding author.

stone) [2, 3]. The Open Stack compute service is known as the nova which is responsible for provisioning instances using the virtualization technologies along with the management of the compute resources [4]. For the organizations that adapt cloud computing the challenges occurred are the security, trust and the privacy since most of the business organizations move their data to the cloud and centralize the management of all the designed data centers, services and the applications [5]. In order to provide the cryptography based security following techniques are analyzed ElGamal and Diffie-Hellman key exchange.

In 1984 Taher ElGamal introduced a cryptosystem [6] which depends on the Discrete Logarithm Problem. The ElGamal encryption system is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie-Hellman key exchange [7]. ElGamal depends on the one way function, means that the encryption and decryption are done in separate functions. It [8] depends on the assumption that the DL can't be found in feasible time, while the reverse operation of the power can be computed efficiently.

The first public key system proposed by Diffie and Hellman requires association of both sides to compute a common private key. This poses issues if the cryptosystem should be applied to communication system where both sides are not able to interact in reasonable time because of deferrals in transmission or inaccessibility of the receiving party [10]. Diffie and Hellman is not a general purpose encryption algorithm as it can only provide secure secret key exchange. Thus [11] it presents a challenge for the cryptologists to design and provide a general purpose encryption algorithm that satisfies the public key encryption standards. So, after Diffie-Hellman [12], RSA public key cryptosystem came. After RSA, The ElGamal solved the Diffie-Hellman key exchange algorithm by presenting a random exponent type *k*. This exponent is a replacement for the private type of the receiving entity. Because of this simplification the algorithm [13] can be utilized to encode in one heading, without the need of the second party to take effectively part. The key development here [14] is that the algorithm can be utilized for encryption of electronic messages, which are transmitted by the method for public store-and-forward services. One of the strengths [15] of ElGamal is its non-determinism-encrypting the same plaintext multiple times will result in different cipher texts, since a random *k* is chosen each time.

Even though [16] these techniques are providing better security, it requires additional platform for implementation. For that purpose SDN is joined with XTR. With SDN, an administrator can change any network switch's rules when necessary-prioritizing, deprioritizing or even blocking specific types of packets with a granular level of control and security. This is especially helpful in a cloud computing multi-tenant architecture, because it enables the administrator to manage traffic loads in a flexible and more efficient manner [17]. Essentially, this enables the administrator to use less expensive commodity switches and have more control over network traffic flow than ever before.

Other benefits of SDN are network management and end-to-end visibility. A network administrator need only deal with one centralized controller to distribute policies to the connected switches, instead of configuring multiple individual devices [18]. This capability is also a security advantage because the controller can monitor traffic and deploy security policies. If the controller deems traffic suspicious, for example, it can re-route or drop the packets. Even though SDN enables optimal sharing of network resources, it lacks in providing security since it is based on the centralized approach, devices involved in networking will not mutually trust each other so packets sending by the authorized node might be captured and altered by the third party. So in order to provide the cryptography based security through SDN, XTR is employed.

The security is identified as a major threat in the Open Stack cloud environment that comprises of some specified components such as Nova, Neutron, Cinder Block store, Horizon *etc.* The security issues mainly tend to occur in the Neutron component. In order to overcome security, issue in neutron component we proposed novel algorithm is the hybrid Auto Sec SDN-XTR (Automated end to end Security in Software Defined Networks – Efficient and Compact Subgroup Trace Representation).In order to rectify the security challenges [19] additionally an efficient security algorithm XTR is proposed for providing the encryption of the file content that also involves a trace operation to incorporate integrity checking. This provides efficient security by involving the Diffie-Hellman for key agreement (both the public key and private key) and ElGamal approach for encryption. Then after the networking process storage of the files content occur in the cinder block store environment. In the cinder store erasure codes algorithm is utilized for data recovery where less storage will be achieved since replicas are not utilized and duplication of file content will not be done instead only parity data will be created as in the concept of RAID (Redundant Array of Independent Disk). Now the unique data which are recovered in cinder block are already been secured by XTR encryption and should be effectively distributed.

The rest of the paper is organized as the related researches in the section 2; the section 3 describes the proposed methodology and results followed by the references.

2. RELATED RESEARCHES

This section provides an overview of the security mechanism for the open stack cloud available in the literature.

Ali *et al.* [19] discussed about cloud computing that was meant to provide cost effective, elastic, easy to manage and powerful resources on the fly, over the Internet. The capabilities of the hardware resources were increased by optimal and shared utilization. The features described encourage the organizations and individual users to shift their applications and services to the cloud. Additional security threats occur when the when some services are provided by the third party cloud providers. All the nature of the security issues identified is detailed. The virtual networks raise as some unique security concern occur in addition to the other security concerns. Due to the lack of administrative control of owner organization identity management and access control problems are recognized over the organization's digital resources also takes distinctive forms in cloud.

Chadwick *et al.* [20] described about the Open Stack which is an open source cloud computing and is responsible for securing federated community clouds, *i.e.* inter-clouds. There are methods for federated identity management (FIM) that permit authentication and authorization to be flexibly enforced across federated environments. The goal of adding protocol independent federated identity management to the Open Stack services was addressed and so a detailed federated identity protocol was presented. The implementation of the protocol independent system components, along with the incorporation of two different FIM protocols, namely SAML (Security Assertion Markup Language) and Keystone proprietary was presented in detail.

Hakiri *et al.* [21] discussed about the issues such as the availability of the network, providing end-to-end connectivity for users and allowing dynamic QoS management of network resources for new applications, such as data center, cloud computing, and network virtualization. To address these problem Software Defined Networking (SDN) concepts was introduced. The separation and centralization of the control plane was identified from the forwarding elements in the network as opposed to the distributed control plane. With the aid of decoupling deployment of control plane software components was allowed on computer platforms that are much more powerful than traditional network equipment while protecting the data and intellectual property of the vendors of such equipment. To address the multiple challenges in realizing the Future Internet and to resolve the ossification problem of the existing Internet. To address these requirements, the wide range of recent and state-of-the-art projects on SDN.

Chen *et al.* [22] described mechanism to protect outsourced data in cloud storage against corruptions, adding fault tolerance to cloud storage, along with efficient data integrity checking and recovery procedures. Regenerating codes provide fault tolerance by striping data across multiple servers, while using less repair traffic than traditional erasure codes during failure recovery. The problem of remotely checking was checked by the integrity of regenerating-coded data against corruptions under a real-life cloud storage setting. A practical data integrity protection (DIP) scheme was designed for a specific regenerating code, while conserving its intrinsic properties of fault tolerance and repair-traffic savings. The DIP scheme designed was developed under a mobile Byzantine adversarial model, and enables a client to feasibly validate the integrity of random subsets of outsourced data against general or malicious corruptions.

Azadeh Tabiban *et al.* [23] designed and implemented a middleware, PERMON, as a pluggable interface to Open Stack for intercepting and verifying the legitimacy of user requests at runtime, while leveraging our previous work on proactive security verification to improve the efficiency and to ensure the accountability of a cloud environment, security policies may be provided as a set of properties to be enforced by cloud providers. However, due to the sheer size of clouds, it can be challenging to provide timely responses to all the requests coming from cloud users at runtime. Nelson Mimura Gonzalez *et al.* [24] analyze the practical security of an Open Stack IaaS cloud when combined with either Open ID Connect (using Google as IdP) or Facebook Connect (using Facebook as IdP). The criteria used in the analysis comprise the ability to provide data encryption, the risks involved in the use of an external IdP, and improper access control. The installation and configuration of cloud environments has increasingly become automated and therefore simple. For instance, solutions such as RedHat RDO and Mirantis Fuel facilitate the deployment of popular computational clouds like Open Stack. Despite the advances in usability, effort is still required to create and manage multiple users. This is of particular relevance when dealing with sensitive information, a somewhat common case for private clouds. To alleviate this burden, many clouds have adopted federated Single Sign-On (SSO) mechanisms for authenticating their users in a more transparent manner.

In the aforementioned related works lot of challenging problems are identified for providing the security of open stack in the cloud management, [18] discussed for identifying the security issues in the network and also it lacks of administrative control of owner organization identity management. [19] Suffers from access control problems to arrange for the end to end security in the network platform in realizing the Future Internet. In order to resolve the problem [20] presented security against the communication path and also Cloud storage, along with efficient data integrity checking and recovery procedures. [21] explained regenerating codes which fails to provide fault tolerance by striping data across multiple servers, while using less repair traffic during failure recovery are identified The above mentioned problems are revised to solve this issue to provide an end to end high security mechanism and to solve the recovery of data during communication by providing an efficient security in the open stack cloud management.

3. IMPROVED SECURITY MECHANISM IN OPEN STACK BASED ON HYBRID AUTO SEC SDN-XTR SECURITY ALGORITHM

Open Stack is a cloud management system that is meant to control a large pool of the compute, storage and networking resources throughout a datacenter managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. The Open Stack cloud includes various problems such as the multitenant nature due to the presence of many components or tenants in the Open Stack environment, vulnerabilities in Open Stack dashboard and other than that it includes the major security threats (CIA) such as the Confidentiality, Integrity and Availability. The major threats identified include the protection of the data during the transmission of the data in the network API *i.e.* the Neutron Open Stack environment, virtualization security occurring during the data replication in the Cinder Block Store Open Stack environment, all this major threats occur due to the open source nature and so open Stack widely adapts some security algorithms that include their own limitations. And hence an efficient algorithm is proposed as mentioned in the proposed methodology.

The security is identified as a major threat in the Open Stack cloud environment that comprises of some specified components such as Nova, Neutron, Cinder Block store, Horizon *etc.* The security issues mainly tend to occur in the Neutron component. In order to overcome security, issue in neutron component we proposed novel algorithm is the hybrid Auto Sec SDN-XTR (Automated end to end Security in Software Defined Networks – Efficient and Compact Subgroup Trace Representation). SDN is utilized for some undertaking in the system joined with steering and security. System association between two endpoint, for example, server are secured by utilizing end to end safety efforts by encryption The mix of end-to-end security and SDN could empower the advancement of automated security measures(Auto Sec SDN) to beat arrange threats. Auto sec SDN is a conclusion to end security calculation in systems administration that conveys security against the internal threats such as the eavesdropping, Denial-of-Service (DoS) and is additionally implied for correspondence with another device through secure administrations or all correspondence from a specific customer/application must be encoded. In any case, during the general task for correspondence just unique associations are secured. Another association isn't secured. On the off chance that Auto Sec SDN-Agent isn't introduced, the assailant will have the capacity to alter information or to listen stealthily on any system; Auto Sec SDN-Protocol additionally effectively speaks with the host without security instrument.

In order to rectify the security challenges additionally an efficient security algorithm XTR is proposed for providing the encryption of the file content that also involves a trace operation to incorporate integrity checking. This provides efficient security by involving the Diffie-Hellman for key agreement (both the public key and private key) and ElGamal approach for encryption. Then after the networking process storage of the files content occur in the cinder block store environment. In the cinder store erasure codes algorithm is utilized for data recovery where less storage will be achieved since replicas are not utilized and duplication of file content will not be done instead only parity data will be created as in the concept of RAID (Redundant Array of Independent Disk). Now the unique data which are recovered in cinder block are already been secured by XTR encryption and should be effectively distributed. The Swift Block Store component recognized for storing and spread the copies of file content to distribute them over different servers so the loss of the data does not cause any effect and a cost effective storage will be done with high availability. Finally, all the security and the recovery works performed will be computed by the nova which is a compute service and all processing performed by nova, Neutron and Cinder Block Store are updated in the Hinder Open Stack component. The proposed work will be implemented on java platform. The overall proposed methodology explained in the Fig. 1.

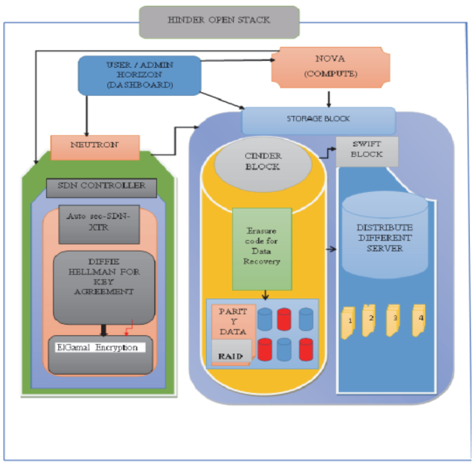


Fig. 1. Open stack proposed component architecture.

Initially the open stack is created in the hinder open stack which controls the large pool of the storage and networking resources through datacenters managed through dashboard that gives the administrator control while authorized their users firstly login page/signup page for web interaction. Open stack administrator uploads the file using dashboard to distribute the data in different server using open stack Component, let us consider the file in the format mathematically.

Let us consider unauthorized file *i.e.* Public data p mathematically written as:

$$p = k^c. \tag{1}$$

Where p is the file name of public data is content of the file.

Let us consider authorized file *i.e.* private data p' mathematically written as:

$$p' = j^c. \tag{2}$$

Where p' is the file name of private data, and j^c is the content of the file.

3.1 Enriched Security Mechanism in the Neutron Component

Software-Defined Networking (SDN) is a potential technology to automate many tasks in the network including routing and security and to reduce configuration effort for switches/routers by utilizing interfaces like Open Flow. Network connections between two endpoints, such as servers or workstations, are secured by using end-to-end security measures such as encryption. The combination of end-to-end security and SDN could enable the development of automated security measures to overcome network threats. This paper proposes Auto Sec SDN, which shows how end-to-end security can be integrated in SDN to improve the overall network security. Explained in the below section.

3.1.1 Auto sec SDN capabilities and installation

The client sends the configuration request to the controller which contains IP, Destination, source *etc.* The controller initiates the installation of the flow rules on the SDN switches; the clients and the server download the security mechanisms from the repository. Now the client and the server can communicate securely as they conform to the security policies. The Auto Sec SDN configuration environment setup explained in Fig. 2.

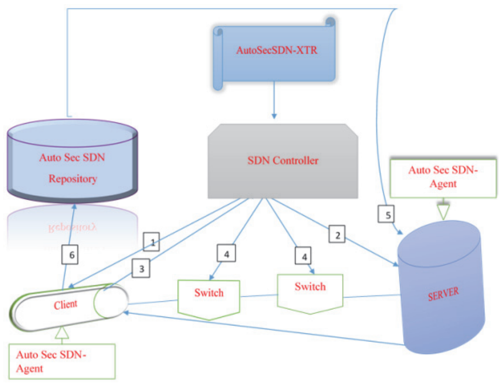


Fig. 2. Auto sec SDN initialization architecture.

A basic setup of Auto Sec SDN is shown in Fig. 1. One aim of Auto Sec SDN is, to keep the configuration effort to a minimum. Therefore, Auto Sec SDN introduces an entity called Auto Sec SDN-Repository, which stores files which are needed by the hosts to fulfill the security policies. An agent, who is called Auto Sec SDN-Agent, is installed on the Client and the Server. It fetches required files from the repository and configures the respective network device. The whole configuration is managed by the Auto Sec

SDN-App which is designed as an extension to the SDN controller. Besides the basic network configuration, Fig. 1 shows how Auto Sec SDN configures itself for securing a connection between the Client and the Server. In this example, the connection attempt originates from the Client, which tries to send the first network packet to the Server. At the same time, it keeps a copy of the network packet for later use. If flow rules of the receiving SDN switch do not match with the network packet, it is intercepted by the agent and is sent as configuration request to the Auto Sec SDN-App in Step 1. The Auto Sec SDN-App searches for the connection details of the network packet in a database.

The database is managed by the administrators of the network and contains the necessary rules of the security policy. The response depends on the result of the database lookup. If no matching entry is found, the Auto Sec SDN-Agent is instructed to forward the network packet without further measures. If an entry is found, the Auto Sec SDN-App instructs the Auto Sec SDN-Agents of both connection participants to load the Agent-Configuration, which is specified in the entry. The Auto Sec SDN-Agents use the provided information to conduct the configuration. This is shown in Step 2 for the Auto Sec SDN-Agent of the Server and in Step 3 for the Auto Sec SDN-Agent of the Client. At the same time the SDN controller installs the appropriate flow rules for the connection on the involved SDN switches by using the Open Flow protocol, as shown in Step 4, which ensures that the connection will be forwarded by the SDN switch. In Steps 5 and 6, the Auto Sec SDN-Agents of the Client and Server retrieve the necessary Agent-Configuration from the Repository.

After loading the Agent-Configuration, the Auto Sec SDN configuration is completed. The Auto Sec SDN Agent processes the copy of the network packet using the new configuration. Then, the network packet is sent to the Server, where the Auto Sec SDN-Agent processes the network packet using its new configuration. After the processing, the network packet is forwarded from the Auto Sec SDN-Agent to the actual Server, which receives the original network packet sent by the Client. The process, which was described before covers the first scenario where the aim is to secure a single connection and the Client initiates the configuration process because it is the origin of the connection. In the second scenario, the Auto Sec SDN-App initiates the configuration process and instructs all network participants to (re)configure. It is necessary that the Auto Sec SDN-App initiates the configuration process after a threat is detected and reported to the Auto Sec SDN App. In this case the first step in Fig. 1 is omitted and the Auto Sec SDN-App sends the instructions immediately to the network participants, as shown in Steps 2 and 3. After this has happened, the configuration of the network is accomplished as described before.

The aim of Auto Sec SDN is to improve the internal network security. The Auto Sec SDN only to secure a single network connection or to secure entire network connections of a host. During regular operation for transmission of data only special connections are secured. But if a security issue occurs, all connections of the network are kept secure regardless of their relevance. If any host is not able to meet the security requirements, for example because Auto Sec SDN-Agent is not installed, it will not be able to communicate with other hosts. In order to solve this problem with a careful configuration of Auto Sec SDN, the attacker will not be able to tamper data or to eavesdrop on any network participant. By using authentication mechanisms for the Auto Sec SDN-Protocol, also the theft of identities and spoofing can be prevented. We proposed an efficient security algorithm XTR for providing the encryption of the file content that also involves a trace operation to incorporate integrity checking This Proposed efficient security algorithm XTR is combined by involving the Diffie-Hellman for key agreement (both the public key and private key) and ElGamal approach for encryption. The efficient security methodology setup is described in the below section.

3.2 Security Algorithm XTR Setup

As a brief introduction, XTR makes use of traces to represent and calculate powers of elements of a subgroup of a finite field. XTR public key data contain two primes x, y and the trace Tr(y) of a generator of the XTR group. It uses subgroup of GF(x^2) arithmetic while achieves by subgroup of GF(x^6) full security leading to substantial savings both in communication and computational overhead without compromising security. With primes x and y of a generator of about 1024/6 170 bits, the security of XTR is equivalent to traditional subgroup systems using 170 bits' subgroups and 1024 finite fields. The improved security mechanism combined by Diffie-Hellman for key generation and ElGamal for encryption technique which is based on XTR algorithm is described below subdivision.

3.2.1 Asymmetric key agreement

Diffie-Hellman key exchange algorithm is utilized to share a public data get in the Eq. (1) and private data get in the Eq. (2) matches for key generation security. The asymmetric key is scrambled utilizing the shared secret key for secure transmission. The public key is guaranteed by the endorsement expert to keep a man-in-the-center attack. Any number of members can participate in secure trades by performing emphases on the understanding convention and trading middle of the road information. Here, two clients, who are obscure to each other offer secret key through a shaky channel. At first, those two offer a public key for validation. The outsider may get to the keys, while transmission, which is generally known as the man in the center attack. It might change the key shared between both the sender and the Receiver.

3.2.2 Diffie-Hellma key exchange algorithm

In Step 1, two figures such as a prime integer G and a producer P is nominated file via together the sender besides the receiver. Then two random numbers x, y that stand fewer than the prime integer G are designated as private keys. The public data (p) get in Eq. (1) the public keys of the sender and private key of the receiver in Eqs. (3) and (4)

$$A = p^x \bmod G, \tag{3}$$

$$A' = p^y \bmod G. \tag{4}$$

These public keys Eq. (3) were swapped between the sender and the receiver via a self-doubting then the Private Key of sender and receiver is in Eqs. (5) and (6)

$$B = (p'^{(y)} \bmod G)^x, \tag{5}$$

$$B' = (p'^{(x)} \bmod G)^y. \tag{6}$$

The shared secret key '*c*' of both sender and receiver is

$$C = (p^{xy} \bmod G). \tag{7}$$

The entire shared secret key exchange of both sender and receiver overall protocol architecture explained in Fig. 3 given below.

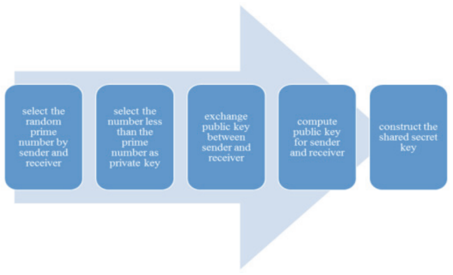

select the random prime number by sender and receiver
select the number less than the prime number as private key
exchange public key between sender and receiver
compute public key for sender and receiver
construct the shared secret key

Fig. 3. Key exchange architecture.

The shared secret key is converted to cipher text which is subjected for additional security encryption of the file using ElGamal approach.

3.2.3 ElGamal approach for encryption

The encryption algorithm works as follows: to encrypt a message *m*, to sender under public key.

The Receiver chooses a random shared key *y* from $\{1.q-1\}$, in Eq. (7) then calculates of random key of the file

$$D = g^y. \tag{8}$$

Then the Receiver maps his message '*m*' on to an element of *D* in Eq. (8).

Then Receiver calculates unique key id using Eq. (7)

$$E = m.C. \tag{9}$$

Finally Receiver sends the encryption cipher text '*F*' to sender using Eqs. (7)-(8)

$$F = C.D.M = gy.(p^{xy} \bmod G).M. \tag{10}$$

Where '*m*' is the file message.

The benefit of this ElGamal approach is that, it makes efficient encryption process. After this process, the asymmetric key (both public and private key) is encrypted by the employment of Diffie-Hellman algorithm (DH) and then the key will be added with the plaintext. The cipher text is generated. The encrypted data is transferred to the receiver. At the receiver side, Inverse process is applied and the same keys are employed in order to recover the unique indication message. Consequently, the hybridization of the technique additionally involves a secure transmission of file content through the network API.

The main advantage of the proposed Algorithm XTR additionally checks the latency occurring in the network by involving the secure shortest path estimation of the nodes in the network rather than not preferring a specified path and thus reducing latency. Only the secure nodes are selected that is not affected by any internal or external attack. During the transmission the integrity of the file content is also checked in the network with the aid of this hybridized security approach and hence any packet loss or integrity problems occurring will be sustained.

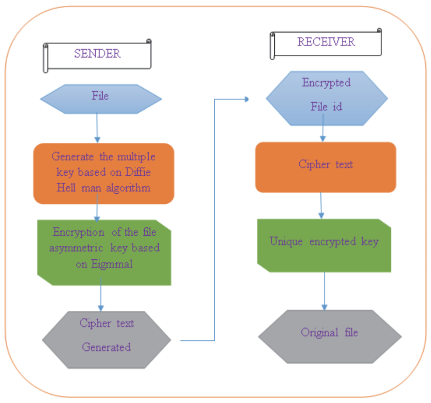

SENDER
File
Generate the multiple key based on Diffie Hell man algorithm
Encryption of the file asymmetric key based on Elgmmal
Cipher text Generated
RECEIVER
Encrypted File id
Cipher text
Unique encrypted key
Original file

Fig. 4. Proposed security mechanism based on Diffie-Hellman and ElGamal approach.

After providing the security mechanism in the networking process the file will be stored in the cinder store, erasure codes algorithm is used for data recovery.

3.3 Erasure Code Mechanism for Data Recovery

After securing the data, which is not affected by any internal or external threats there is a chance for data loss during transmission. This lead to a necessity of data recovery approach which should recover the data effectively even if it find duplicates. This is achieved by the usage of Erasure code mechanism for recover then data.

- Erasure codes utilize less capacity than imitations.
- Rather than copying the total question numerous circumstances, equality information is made much like Strike.
- This can lessen the measure of capacity for a question from 3X to 1.2X.
- Erasure coding is an answer for financially savvy flexibility and high accessibility.

Given a unique information record, deletion code initially separates it into *k* parts of a similar size and after that encodes them into n sections. Any *k* pieces removed from then encoded ones can be utilized to reproduce the first information document. In the meantime, it is difficult to acquire any data about any *k* piece of the first information from not as much as parts. In this manner, deletion code underpins *k*-protection and guarantees high security. This code is generally alluded to as erasure code. Mathematically, (n, k) erasure code can be expressed as

$$S = V.F. \tag{11}$$

In the above equation $F = \{F_i \mid 1 \le i \le k\}$ is the original data file retrieved in the Eq. (10), F is the encoded data file, $s = F = \{s_i \mid 1 \le i \le n\}$. The Improved Erasure Code for Information Recuperate Clarified In the Well-ordered in the given Fig. 5.

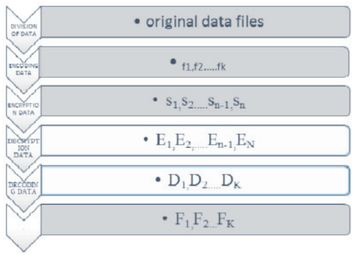


Fig. 5. Enhanced erasure code-based recovery mechanism.

In the wake of making the erasure code for information recuperation where less capacity will be accomplished since replicas are not used and duplication of the document substance won't be done rather just equality information will be made as in the idea of Strike The utilization of equality bits is a typical technique for recognizing mistakes in information transmission and storage. Before taking a glance at the utilization of equality bits in the Attack, we should look all the for the most part at their utilization as a strategy for blunder identification.

3.3.1 Parity data generated using raid concept in cinder block

After successfully recovering the file using erasure code, the less storage will be achieved since copies are not exploited and duplication of the file content will not be done in its place only parity data is generated based on the concept of RAID. In data transmission, if data is sent starting with one device then onto the next with no type of mistake rectifying system, the Receiving device must expect the data is right. The quantity of blunders amid data transmissions is much lower with computerized transmission than when the simple transmission was pervasive. In any case, data transmission is once in a while totally mistake free, so it is rash to accept that data got is precisely the same as was transmitted. The utilization of an equality bit is a method for including checksums into data that can empower the objective device to decide if the data has been gotten effectively.

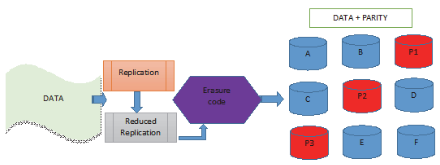


Fig. 6. Data storage mechanism in cinder block.

After creating the parity datum using RAID concept the stored transmission file shifted to the Swift Block to Store part perceived for putting away subsequently spread the duplicates of record substance to circulate them over various servers so the loss of the information does not cause any impact and a financially savvy stockpiling will be finished with high accessibility.

3.4 Allocate the Copy of the File with the Different Server Using Swift Component

The swift block store component recognized for storing and spreading the copies of the file content. This will distribute over the different server so loss of the data does not cause any effect and cost effective storage will be evolved with high availability. Swift is a two-level storage framework comprising of an intermediary level, which handles every single approaching request, and a question stockpiling level where the real information is put away. Swift uses an information structure called a ring to layout URL for a question a specific area in the bunch where the protest is put away.

3.4.1 Swift data model architecture

For one of a kind storage area, Open Stack Swift enables clients to store unstructured data objects with a sanctioned name containing three sections: Ring, Disk, and object. The initial segment Ring storage area is an interestingly named storage territory that contains the metadata (spellbinding data) about the Ring itself and the rundown of Disks in the Ring. The second part Disk storage area is the client characterized storage region inside a Ring where metadata about the Disk itself and the rundown of objects in the Disk will be put away. What's more, last part object storage area is the place the data object and its metadata will be put away. Data arrangement has a chain of importance of Regions, zones, servers and drive when the client requests for administrations storage bunch need to find data and it will take a gander at the storage area (/Ring, /Ring/Disk, /Ring/Disk/object) and counsel one of the three rings: Ring, Disk ring or object ring. In our framework the transmission of the document spread over the disk in to the distinctive server clarified in the diagrammatically in Fig. 7.

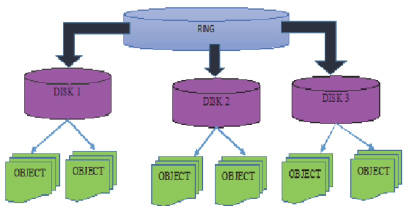


Fig. 7. Storage location overview in swift.

Swift component recognized for storing thus spreading the copies of fie content to distribute them over different server so loss of the data does not cause any effect and a cost effective storage will be done with higher availability finally all the security and the recovery works performed will be computed by nova which the computing service and all processing performed by nova, neutron, and cinder block store are updated in the hinder open stack component.

4. RESULT AND EVALUATION

The experimental open stack is deployed following a four node configuration: compute node, network node, storage block node and swift node proposed system for providing the security to measure is identified between the networks in neutron component and cinder component it is used to implement in the working platform of JAVA. The system configuration is given as

Processer: Intel(R) Pentium(R) CPU G2030 @ 3.00GHz 3.00GHz
Operating system: Windows 7 Ultimate
System type: 64-bit operating System
Implementation tool: Net Beans IDE 7.4
Language: Java
Java Additional package (API):

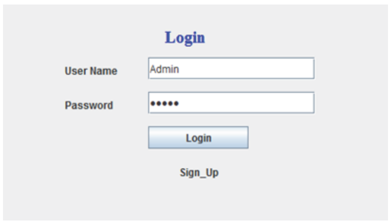
cloud creation	Openstack-swift-1.5.0
	Swift-1.6.0-rc. 1-sources
	Openstack-nova-1.5.6
	Openstack-keystone-1.5.1
	Openstack-cinder-1.6.1
	Jclouds-slf4j-1.6.0-rc.4
	Jclouds-core-1.6.0-rc.1
	com.google.guava-1.6.0
database storage	My sql-connector
graph	jfreechart-1.0.6

4.1 Simulation Result of Proposed Component

Initially the open stack is created in the hinder open stack which controls the large pool of the storage and networking resources through datacenters managed through dashboard that gives the administrator control while authorized their users firstly login page/signup page for web interaction. The security issue is the major thread problem in the open stack in order to overcome our proposed system we propose the auto sec SDN-XTR algorithm for end to end security while transferring the file in order to give the additional security the encryption methodology is proposed in the neutron component. The proposed algorithm estimates the key generation and encryption of the file content that also involve trace operation to incorporate integrity checking. After file shared in network API the files are stored in the cinder storage block in order to recover the data the proposed methodology implement the ensure code algorithm is utilized. Duplication file content will not allow so parity data will be generated with the concept of RAID, the copies of the data will be distributing them over the different server so loss of the data does not cause any effect and cost effective storage will be done with the high availability finally all the security and recovery works performed will be computed by nova.

4.1.1 Implementations of horizon/dashboard for control open stack

The dashboard is generally installed on the controller node. Cloud administrator can access the dashboard, and for user register user name and password. On the Log in page, enter user name and password, and click Sign. In the top of the window displays user name. You can also access the sign out of the dashboard. If you are logged in as an end user, the open stock component blocks are displayed. If you are logged in as an administrator, the component tab and admin tab are displayed. Project components are organizational units in the cloud and are also known as tenants or accounts. Each user is a member of one or more projects. Within a project, a user creates and manages instances since Open Stack dashboard is responsible for maintaining other Open Stack components such as the Nova, Neutron and Cinder Block Store. The neutron components are described in the below section which is used for networking between other applications.


Login

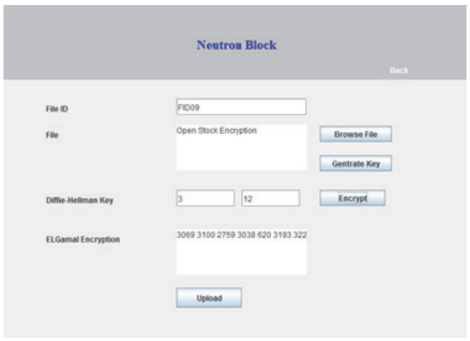
User Name: Admin

Password: •••••

Fig. 8. Dashboard gives administrator control for login/signup page.

4.1.2 Implementation of neutron block for communication in the open stack

The neutron blocks which is used for communication between the one end to another end in this Fig. 9 reprsents file transmission between the one end to another end to tackle the security mechanism using cryptology method. Initially the file is insered in the source which can access by admin/user. After inserting the file Diffie-Hellman approch is used to generate key,which is automatically generated depends on file size after creating the key. In order to give the additional security we give ElGamal approch for encryption method for highly secured communication.The main advantage encryption approch is, it additionally involves a secure transmission of file content through the network API. The key generation time and encryption times are plotted in the below section.


Neutron Block

File ID: FID09

File: Open Stock Encryption

Diffie-Hellman Key: 3 12

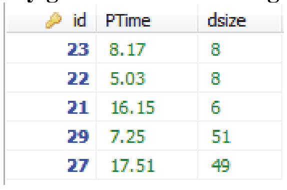
ELGamal Encryption: 3069 3100 2759 3038 620 3193 322

Fig. 9. Neutron component providing security using XTR-algorithm.

4.1.3 Key generation time in neutron block

A remote channel is described by its two end clients. By transmitting pilot flags, these two clients can gauge the channel amongst them and utilize the channel data to produce a key which is a mystery just to them. The regular mystery key for a gathering of clients can be produced in light of the channel of each match of clients.

Table 1. Secret key generation time using Diffie-Hellman.



id	PTime	dsize
23	8.17	8
22	5.03	8
21	16.15	6
29	7.25	51
27	17.51	49

In Table 1 secret key generation randomly using Diffie-Hellman with different file and processing time is in Table 1. Here id 23 and 22 having the file size 8, which has the secret key generation time 8.17 and 5.03. The file id 21 having the file size 6, which has the secret key generation is 7.25 and also the id 29 and 27 the size of the key is 51 and 49 the processing time of generating secret key is 7.25 and 17.51. And the performance of the result is shown in below graph.

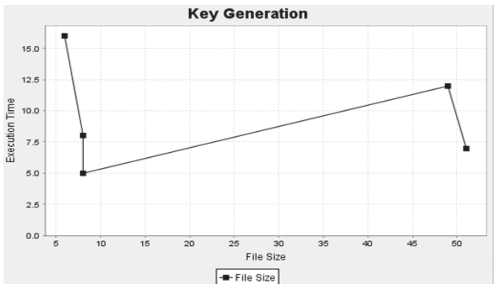


Fig. 10. Key generation time with different files.

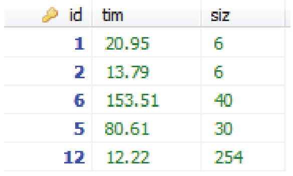
4.1.4 Encryption time generation in neutron block

The encryption time of packet sent is the time calculated during encryption *i.e.* during conversion of plain text to cipher text by high level security algorithm. The encryption time is calculated by using the formula

$$time = \frac{datasize}{speed}.$$

Table 2 explained the encryption time using ElGamal approach with different files and the processing time mentioned in the table. Here id 1 and 2 having the file size 6, which has the encryption time 20.95 and 13.79. The file id 6 having the file size 40, which has the encryption time is 153.51 and also the id 5 and 12 the file sze is 30 and 254 the processing time of generating encryption is 80.61 and 12.22. And the performance of the result is shown in below graph.

Table 2. Encryption time using ElGamal.



id	tim	siz
1	20.95	6
2	13.79	6
6	153.51	40
5	80.61	30
12	12.22	254

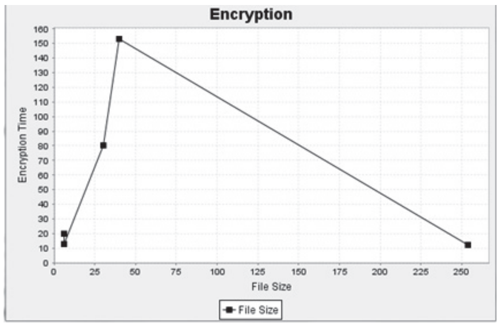


Fig. 11. Encryption time with different files.

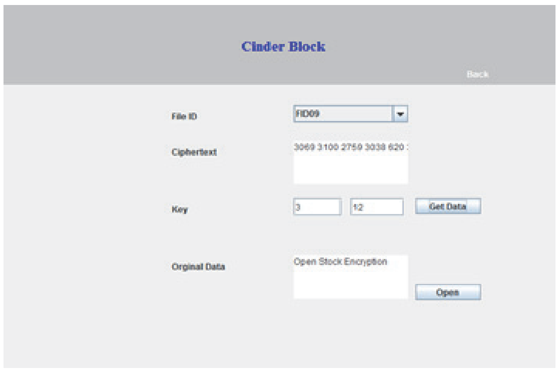
The security issue of the neutron components are rectified by using the proposed algorithm Autosec SDN-XTR that delivers security against the internal threats such as the eavesdropping, Denial-of-Service (DoS) and is also meant for communication with other device through secure services. And gives the better Additional security an efficient security algorithm XTR is used to solve the proposed method additionally checks the latency occurring in the network by involving the secure shortest path estimation of the nodes in the network rather than not preferring a specified path and thus reducing latency. Only the secure nodes are selected that is not affected by any internal or external attack. During the transmission the integrity of the file content is also checked in the network with the aid of this hybridized security approach and hence any packet loss or integrity problems solved.

Then after the networking process storage of the files content occurs in the cinder block store environment which describe in Fig. 12.

4.1.5 Implementation of cinder block for storage of the file in the open stack

User can view the storage file in this block. In the cinder store erasure codes algorithm is utilized for data recovery where less storage will be achieved since replicas are not utilized and duplication of file content will not be done instead only parity data will be created as in the concept of RAID.

Fig. 12 describes the user selecting the file id which shows only encrypted data and this block collect the private and public key from user. Then decrypt the cipher text using two keys (private and public). Finally, original data will be shown in the concern block. The Swift Block Store component is recognized for storing thus spread the copies of file content to distribute them over different servers so the loss of the data which does not cause any effect and a cost effective storage will be done with high availability. Finally, all the security and the recovery works performed will be computed by the nova which is a compute service and all processing performed by nova, Neutron and Cinder Block Store are updated in the Hinder Open Stack component.



Cinder Block

Back

File ID: FID09

Ciphertext: 3069 3100 2759 3038 620 :

Key: 3 | 12 | Get Data

Orginal Data: Open Stock Encryption

Open

Fig. 12. cinder block providing data recovery using erasure code techniques.

4.2 Comparison Result

4.2.1 Comparison strategy of key generation

The efficiency of proposed methodology is compared with the existing algorithm with the parameters such as key generation time, encryption time. The key agreement time comparison is given for the proposed high level security mechanism along with existing security key generation algorithm such as Adaptive Ant-Lion optimization (AALO), RSA. Since El-Gamal crypto system needs less key bits, it has taken less time for key generation when compared with adaptive ant-lion optimization and RSA. The comparison result is explained diagrammatically in Fig. 13.

Table 3. Secret key generation.

Algorithm	Secret key Generation Time
Proposed Algorithm	0.14
Adaptive Ant-Lion Optimization(AALO)	9.8
RSA	0.27

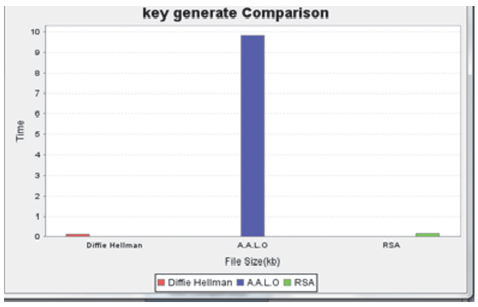


Fig. 13. comparison strategy of proposed method, AALO and RSA.

4.2.2 Comparison strategy of encryption time

The encryption time comparison is given for the proposed authenticate security mechanism along with existing security key generation algorithm such as Adaptive Ant-Lion optimization (AALO), elliptive curve – Diffie-Hellman. The comparison result is explained diagrammatically in Fig. 14.

Table 4. Encryption time.

Algorithm	Encryption Time
Proposed Algorithm	0.284
Adaptive Ant-Lion Optimization (AALO)	0.5201
Elliptic Curve-Diffie Hellman (EC-DF)	0.367

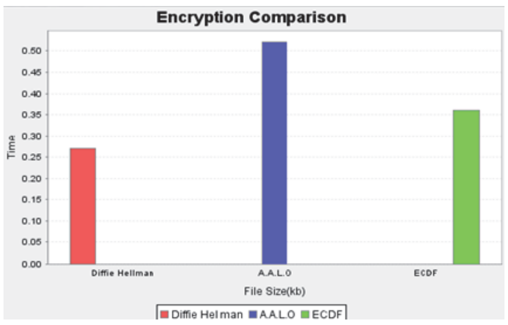


Fig. 14. Comparison strategy of proposed method, AALO and Adaptive Ant-Lion Optimization (AALO).

The key agreement and Encryption comparison strategy between the proposed method and different algorithm are analyzed in the graph in Figs. 14 and 15. The key agreement algorithm such as proposed algorithm of Diffie-Hellman key agreement time value is 0.14, Adaptive Ant-Lion Optimization (AALO) key agreement time value is 9.8 and RSA key processing time value is 0.27, the comparison of AALO and RSA key agreement algorithm the RSA has attain minimum processing time when compared with AALO, RSA compared with proposed method. Our proposed method attain minimum processing time complexity, in overall comparison of proposed method, AALO and RSA attain the best time complexity of key agreement, since El-Gamal crypto system is adapted with the Diffie-Hellman key exchange it provides end to end secure transferring through web interface from source to destination over the public channel so user who have the secret key can only decrypt the file. Thus the proposed system have operated on a web interface derived from the software defined network it is compatible with modern communication.

The encryption time comparison between the different algorithms such as proposed method ElGamal encryption time value is 0.284 with AALO encryption time value is 0.5201 and Elliptic Curve-Diffie Hellman (EC-DF) encryption time value is 0.367. the comparison strategy of EC-DF attains the minimum encryption time when compared with AALO, the overall comparison strategy of encryption time proposed method attain the minimum encryption time when compared with AALO and EC-DF.

5. CONCULSION

During communication it was imperative to protect the data before the process of transferring it. Since if data was hacked by third parties internally then there was a chance of maltreating the data/information. To avoid that efficient proposed novel algorithm was introduced Auto Sec SDN-XTR. Efficient security algorithm XTR proposed for encryption of the file using Diffie-Hellman for key agreement and ElGamal approach for encryption. This approach achieves secret key generation time of 0.14 secs, encryption time of 0.284 secs which are much lesser compared to AALO, EC-DH and RSA. All the security and the recovery worked performed well and propose methodology gave the better time complexity for key agreement and encryption time during data transmission in the neutron component with high security.

REFERENCES

1. M. Lamanna, "Large-scale data services for science: Present and future challenges," *Physics of Particles and Nuclei Letters*, Vol. 13, 2016, pp. 676-680.
2. Y. Yamato, Y. Nishizawa, S. Nagao, and K. Sato, "Fast and reliable restoration method of virtual resources on Open Stack," *IEEE Transactions on Cloud Computing*, Vol. 6, 2018, pp. 572-583.
3. S. Sotiriadis and N. Bessis, "An inter-cloud bridge system for heterogeneous cloud platforms," *Future Generation Computer Systems*, Vol. 54, 2016, pp. 180-194.
4. D. A. B. Fernandes, L. F. B. Soares, J. V. Gomes, M. M. Freire, and P. R. M. Inácio, "Security issues in cloud environments: a survey," *International Journal of Information Security*, Vol. 13, 2014, pp. 113-170.
5. V. Chang, Y.-H. Kuo, and M. Ramachandran, "Cloud computing adoption framework: A security framework for business clouds," *Future Generation Computer Systems*, Vol. 57, 2016, pp. 24-41.
6. A. Corradi, M. Fanelli, and L. Foschini, "VM consolidation: A real case based on Open Stack Cloud," *Future Generation Computer Systems*, Vol. 32, 2014, pp. 118-127.
7. D. Nguyen, J. Park, and R. Sandhu, "Adopting provenance-based access control in open stack cloud iaas," in *Proceedings of International Conference on Network and System Security*, 2014, pp. 15-27.
8. D. Dongre, G. Sharma, M. P. Kurhekar, U. A. Deshpande, R. B. Keskar, and M. A. Radke, "Scalable cloud deployment on commodity hardware using Open Stack," *Advanced Computing*, *Networking and Informatics*, Vol. 2, 2014, pp. 415-424.
9. V. Chang, M. Ramachandran, Y. Yao, Y.-H. Kuo, and C.-S. Li, "A resiliency framework for an enterprise cloud," *International Journal of Information Management*, Vol. 36, 2016, pp. 155-166.
10. J.-M. Kim, H.-Y. Jeong, I. Cho, S. M. Kang, and J. H. Park, "A secure smart-work service model based Open Stack for Cloud computing," *Cluster Computing*, Vol. 17, 2014, pp. 691-702.

11. X. Zhang, T. L. Yang, C. Liu, and J. Chen, "A scalable two-phase top-down specialization approach for data anonymization using map reduce on cloud," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, 2014, pp. 363-373.
12. Y. Yamato, M. Muroi, K. Tanaka, and M. Uchimura, "Development of template management technology for easy deployment of virtual resources on Open Stack," *Journal of Cloud Computing*, Vol. 3, 2014, p. 7.
13. K. J. Grewal, "ElGamal: Public key cryptosystem," Master of Science, Math and Computer Science Department, Indiana State University, 2015.
14. J. Vilaplana, F. Solsona, I. Teixidó, J. Mateo, F. Abella, and J. Rius, "A queuing theory model for cloud computing," *The Journal of Supercomputing*, Vol. 69, 2014, pp. 492-507.
15. M. J. A. Calero and J. G. Aguado, "MonPaaS: an adaptive monitoring platformas a service for cloud computing infrastructures and services," *IEEE Transactions on Services Computing*, Vol. 8, 2015, pp. 65-78.
16. Y. Yamato, Y. Nishizawa, S. Nagao, and K. Sato, "Fast and reliable restoration method of virtual resources on Open Stack," *IEEE Transactions on Cloud Computing*, Vol. 6, 2018, pp. 572-583.
17. P. Korambath, J. Wang, A. Kumar, L. Hochstein, B. Schott, R. Graybill, M. Baldea, and J. Davis, "Deploying kepler workflows as services on a cloud infrastructure for smart manufacturing," *Procedia Computer Science*, Vol. 29, 2014, pp. 2254-2259.
18. M. Díaz, C. Martín, and B. Rubio, "State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing," *Journal of Network and Computer Applications*, Vol. 67, 2016, pp. 99-117.
19. M. Ali, U. S. Khan, and V. A. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Information Sciences*, Vol. 305, 2015, pp. 357-383.
20. W. D. Chadwick, K. Siu, C. Lee, Y. Fouillat, and D. Germonville, "Adding federated identity management to open stack," *Journal of Grid Computing*, Vol. 12, 2014, pp. 3-27.
21. A. Hakiri, A. Gokhale, P. Berthou, C. D. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," *Computer Networks*, Vol. 75, 2014, pp. 453-471.
22. C. H. Chen and P. C. Lee, "Enabling data integrity protection in regenerating-coding-based cloud storage: Theory and implementation," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, 2014, pp. 407-416.
23. A. Tabiban, S. Majumdar, L. Wang, and M. Debbabi, "PERMON: An OpenStack middleware for runtime security policy enforcement in clouds," in *Proceedings of IEEE Conference on Communications and Network Security*, 2018, pp. 1-7.
24. C. G. Batista, C. C. Miers, P. G. Koslovski, A. M. Pillon, N. M. Gonzalez, and A. M. Simplicio, "Using externals IdPs on OpenStack: A security analysis of OpenID connect, Facebook connect, and OpenStack authentication, " in *Proceedings of IEEE 32nd International Conference on Advanced Information Networking and Applications*, 2018, pp. 920-927.

S. Vatchala received the B.Tech degree in Information Technology from Roever Engineering College, Anna University, Chennai. M.E in Computer Science and Engineering from the Srinivasan Engineering College, Anna University, Chennai. Pursuing Ph.D. in Computer Science and Engineering from the Srinivasan Engineering College, Anna University, Chennai, India. Presently, working as an Assistant Professor in the Department of Computer science and Engineering at Dhanalakshmi Srinivasan Engineering College, Perambalur, India. She had 7 years of experience. She presented 10 papers in various international Conference national Conferences.

S. Ravimaran is Principal of M.A.M. College of Engineering, Tiruchirappalli. He is actively engaged in transforming this Institute to the future requirements through strategies and initiatives, adopting disruptive technology route. He is leading the team incorporating changes in academic curriculum, delivery and evaluation techniques. Dr. Ravimaran Shanmugam completed Computer Science and Engineering from Regional Engineering College, Tiruchirappalli, and did his post graduation and doctorate degrees at the Anna University, Chennai in Information and Communication Engineering.

A. Sathish completed (U.G.) B.Tech in Sri Angalamman College of Engineering, (P.G) M.E Network Engineering at Arulmigu Halasalingam College of Engineering. He had 14 years of experience in teaching field and working as an Associate Professor in M.A.M College of Engineering, Trichy. He has presented more than 15 papers in national/international conference.