International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 654–670, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-052

ADRCN: A Framework to Detect and Mitigate Malicious Insider Attacks in Cloud-Based Environment on IaaS

Priya Oberoi
M. M. Institute of Computer Technology and Business Management
Maharishi Markandeshwar (Deemed to be University), Ambala, Haryana, India
D. A. V. Centenary College, Faridabad, Haryana, India
Corresponding author: priya.hrt@gmail.com

Sumit Mittal
M. M. Institute of Computer Technology and Business Management
Maharishi Markandeshwar (Deemed to be University), Ambala, Haryana, India
E-mail: sumit.mittal@mmumullana.org

Rajneesh Kumar Gujral
M. M. Engineering College
Maharishi Markandeshwar (Deemed to be University), Ambala, Haryana, India
E-mail: drrajneeshgujral@mmumullana.org

(Received November 15, 2018; Accepted February 27, 2019)

Abstract
Security is a critical factor for any of the computing platforms. Cloud computing is a new computing environment but still, its basic technology is the Internet. Thus, Cloud computing environment not only has the threats of its own but it is also prone to security issues of its underlying technology i.e. Internet. In this paper, the authors are proposing a secure routing framework viz. Authenticated Dynamic Routing in Cloud Networks (ADRCN) to mitigate the malicious insider attacks while maintaining the path integrity in the Clouds. Symmetric cryptography with hashing is used to maintain the integrity of the path between the source and destination. The purpose of ADRCN is to maintain the integrity of the path between the client and data center. If malicious insider tries to perform an attack between the client and the data center then it will be detected. This work aims to give a solution for detection and prevention of malicious insider attacks in Cloud-based environments.

Keywords-Cloud security, Path integrity, Internal attacks, Malicious insider attacks, Malicious insider attacks detection.

1. Introduction

The Cloud computing and big data are future of the software service model (Tamura, 2017). Although Cloud computing is a new computing environment, the basic technology i.e. Internet is intact. As the Cloud environment is developed using the Internet, the security issues of the Internet are also posed by the Cloud. In Clouds, both the client and provider reside at different geographical locations and data is accessed using the virtual machines through the Internet (Dewangan et al., 2019). In spite of using security measures at the Cloud end transmission of data is still done using Internet technology. Security is always a matter of concern in any computing environment (Kopachevsky et al., 2016).The Clouds are vulnerable to its own threats as well as to the Internet threats. Thus, sophisticated methods are needed to secure the Clouds. The Cloud-based environments require protocols for secure data transmission (Armbrust et al., 2010; Jain and Kumar, 2014; Oberoi and Mittal, 2018).

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 654–670, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-052

Security has three factors namely Confidentiality, Integrity, and Authentication (CIA). The degree of an effect to the confidentiality, integrity and availability because of the abuse of vulnerability influences the security of the entire framework (Bhatt and Anand, 2017). In the field of security, authentication is the assurance that only the authorized users can access the data. Among the several challenges of the Cloud environment, path integrity significantly needs consideration. The latter assures that the quality of data high is correct and unmodified. Verification of path integrity at an untrusted server is the biggest concern of Clouds. Integrity requirements of data need to be modified or discarded without the authorization. Security in terms of integrity is a most important aspect in the Cloud computing environment (Oberoi and Mittal, 2018; Giri and Gaur, 2015; Jain and Kumar, 2016). Therefore, to achieve and maintain the authentication and path integrity of the Cloud-based environments Authenticated Dynamic Routing in Cloud Networks (ADRCN) is proposed in this work. It uses the symmetric algorithms with message authentication code (MAC) algorithm.

Authenticated Dynamic Routing in Cloud Networks (ADRCN) is a secure dynamic routing framework for communication between the client and the data center (DC). This framework provides a mechanism for authenticating the path of communication between the client and the data center (DC) at the same time maintaining the integrity of the path. The proposed framework detects and prevents the malicious insider attacks in the Cloud-based networks.

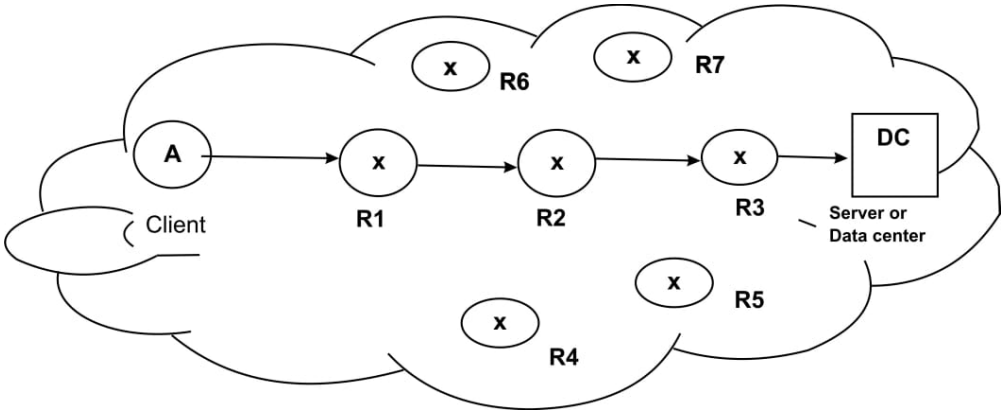

Figure 1. Client (A) communicating with data center (DC)

As the client sends the request to the data center (Figure 1) using the underlying Internet, probabilities are there that the devices in between the path of client and data center (DC) may malfunction. The various intermediate devices may act as Malicious Insiders to take advantage of the network. ADRCN ensures that the path used to send a request from the client to the data server remains the same while getting back the route reply from DC. In general, if the route establishment request from the client A to the data server travels by the path R1->R2-> R3 then the route reply from the destination travels back to the client A by the reverse path i.e. R3-> R2-> R1.

The security for the Clouds has been built with authenticating path and maintaining the integrity of the path. Hashing algorithm (MD5 or SHA-1) with the MAC code is used for this purpose. The client and the data center have the shared key which is generated using the Diffie Hellman algorithm. In this work, the aim of ADRCN is to detect the malicious insider attacks while maintaining the path authentication and path integrity in the Cloud-based environment. ADRCN uses the limited flooding for shared key distribution in the network. The client, the receiver and the intermediate devices are authenticated by the secret key, hashing, and MAC code.

2. Literature Review

A number of methods are developed to maintain integrity of data. A remote attestation model via the Trusted Platform Module (TPM) ensures the integrity and confidentiality of the Clouds (Khan et al., 2011). This model utilizes the trusted Eucalyptus which guarantees that its virtual machines can run only on Cloud nodes with valid integrity. A Cloud-based onion routing system with low latency is capable to provide security from blocking and confronting censors (Jones et al., 2011). This method is effective in terms of connectivity, capacity, and economy of scale. It doesn't solve the problem of the bootstrapping inherent to many anonymity systems. The authors anticipate their future work towards making a system secure from bootstrapping problem.

Information leakage can also prevented by FBCrypt (Egawa et al., 2012). The authors anticipated a method to secure the IaaS. A virtual machine monitor (VMM) is used for encrypting the inputs and outputs between the virtual network computing client and virtual machine (VM) user. The inputs encrypted by the VMM client are decrypted by the VMM when a user VM needs to access it. After updating the frame buffer, the updated pixel data is encrypted by the VMM. This method maintains the authenticity as well as the integrity of the data by using the FBCrypt. The FBCrypt uses the remote attestation of the VMM by a trusted server. RSA key Exchange Protocol is projected by Leena and Rao (2012) to access data securely. The method provides security between the Cloud user and service provider. This solves the problem of key distribution and management. In this method, two-factor authentication methods are used for generating a key using TORDES algorithm. The security is achieved at minimal cost and effort. The RSA algorithm for data security ensures that only the authorized person can gain access to the data and in the case of intrusion, the intruder is not able to decrypt the data and get the original data back (Kalpana and Singaraju, 2012). Nafi et al. (2012) presented a security mechanism for Clouds. This mechanism uses AES file encryption system, RSA to secure communication process, OTP for authenticating the users and MD5 hashing for information hiding. In this method execution time is quite high due to the use of multiple servers for different algorithms. The method mitigates from intruders as it is not possible to gain access to all the servers at the same time. Moreover, decision making is easy due to the use of different servers.

The SHA-2 algorithm for verification of the data integrity together with the XOR method, station-to-station key protocol for generation of keys and mutual authentication with TPA are collectively used to combat the threat of integrity in the Clouds (Kaur and Singh, 2013). A multilayer framework has been presented by Chandramohan et al. (2013). The framework aims for preventing the digital data loss and preserve the secrecy of Cloud. Onion and Garlic privacy layers are used for privacy preserving. This method is capable to mitigate the privacy attacks done by the novel users or unknown users. It has good support for encryption and decryption methods. The future work involves the use of three factors encryption for Clouds.

Kavuri et al. (2014) gave an algorithm which encrypts each file of the third party Cloud server by producing the hash value of 512 bit. It is used to mitigate the security issues of commercial untrusted Cloud servers. The authenticated user accesses the files by proving its identity along with the hash value of message integrity. The applicability of the method was verified using an attributing based encryption process. This model is better than the existing models in terms of time and file size.

Sirisa and Hiranmayee (2015) presented a solution to data security, authentication and integrity problems in their work. The proposed system is the combination of symmetric as well as the hash algorithm. The model is implemented at the client end which uses the combination of Blowfish, RSA and hash value. The client generates the hash value of the data that is stored in its own local safe repository. After which the file is sent to the Cloud for storage. The client is able to check the integrity of the file by computing the hash on the data of the file and comparing it with the previously computed hash value. Mismatch in the two hash values will reflect that data in the file has been changed and thus data integrity is compromised. Authors claim that this method is simple than the third party auditor (TPA) method.

The data anonymization techniques for Clouds is proposed as framework to preserve the privacy along the capacity to perform security and forensic analytics (Dara et al., 2016). Onion modification method is used for multi-layer encryption which is done by increasing the layers of the onion. Hybrid encryption and one way Hash Key chains are used and all the routing servers are allocated keys. The chain is built in the opposite direction. The authors claim that the proposed method is secure and efficient. A framework is presented by Gor and Jain (2016), which has more than two encryption layers. Onion encryption is used to increase the database security and confidentiality during the transmission of data to/from the database. An unauthorized person is not able to retrieve the data. Algebraic signatures are also presented as a solution to Cloud data integrity by Shen et al. (2017). The authors verify that this system is capable to support the data dynamics.

In the work by Patil and Rai (2018) a system to maintain the integrity of the data has been proposed. The authors used the AES algorithm, SHA -512 and Merkle Hash Tree (MHT). A TPA (Third party auditor) is used to keep a check of data integrity. This system reduces the server computation time, audits dynamically and provides dynamics data operation. In the similar work by Chen et al. (2018) a repetitive method to verify the integrity of data on a Cloud has been proposed. The method is MAC based and is capable to resist replay and Man-in-the-Middle (MIMA) attacks. Remote Data Possession (RDP) method is also used by Hariharasitaraman and Balakannan (2018) to maintain integrity in Clouds. The authors used the Fractal tree to organize data in files, which in turn consists of blocks of data. For every block of data, a tag is generated and a mathematical function based on hashing is added to each tag. This hash value guarantees the tag integrity. This method has reduced the complexity of storage and reasonable complexity of computation and communication. This method can be used to support data dynamics under a multi-Cloud environment as future work. Table 1 lists the various techniques given by different authors for maintaining the security and data integrity.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 654–670, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-052

Table 1. Works related to data security and integrity

Authors	Focus on	Technique	Features
Khan et al. (2011)	Integrity and Confidentiality	Remote Attestation Model	Practical in terms of Performance
Jones et al. (2011)	Cloud Security	Onion Routing System	Effective Connectivity, Capacity and Economy of Scale
Egawa et al. (2012)	Data Authentication and Integrity at IaaS	FBCrypt	Remote Attestation
Leena and Rao (2012)	Security in Clouds	TORDES Algorithm	Minimal Cost and Effort
Nafi et al. (2012)	Multilevel Security	AES , RSA , OTP and MD5	Easy Decision making, Mitigates Intruders
Kalpana and Singaraju (2012)	Data Security	RSA	Mitigates Intruders
Kaur and Singh (2013)	Data Integrity in Clouds	SHA-2, Station-to-station key protocol, Mutual authentication with TPA	Ensure Data Integrity
Chandramohan et al. (2013)	Digital Data Loss and Secrecy of Cloud	Onion and Garlic Encryption	Good Support for Encryption and Decryption
Kavuri et al. (2014)	Security in Clouds	Hash Value of 512 bit	Better in terms of Time, File Size
Sirisa and Hiranmayee (2015)	Security, Authentication and Integrity	Blowfish, RSA, Hash Value	Simple than Third Party Auditor (TPA)
Dara et al. (2016)	Security for Clouds	Onion Encryption, Hybrid Encryption, One way Hash Key chains	Secure, Efficient
Gor and Jain (2016)	Database Security and Confidentiality	Onion Encryption	Secure, Better Security
Shen (2017)	Data Integrity in Clouds	Algebraic Signatures	Supports Data Dynamics
Patil and Rai, 2018	Data Integrity	AES Algorithm, SHA - 512 , Merkle Hash Tree (MHT).	Reduced Server computation Time, Dynamic Audit, Dynamic Data Operation
Chen (2018)	Verify Data Integrity in Clouds	MAC	Resist Replay and Man-in-the-Middle(MIMA) attacks
Hariharasitaraman and Balakannan (2018)	Data Integrity in Clouds	Remote Data Possession (RDP), Fractal Tree, Hash Value	Reduced Storage complexity ReasonableComputation and Communication Complexity

3. Types of Attacks in Clouds

Broadly the attacks in Cloud-based environment are classified as External attacks and Insider attacks(Boppana and Su, 2007; Oberoi and Mittal, 2017;).

A. *External Attacks:* The attacks in which the aim of the attacker is to cause congestion, propagate fake routing information and disturb the VMs or the connecting devices from providing services are called external attacks. External attacks in the Clouds are similar to external attacks in traditional computing or networking environments. These attacks can be effectively prevented and detected by the techniques like – firewall or authentication etc..

B. *Insider/ Internal Attacks*: Due to the invasive nature and open platform in the Clouds, the insider attacks are more harmful and dangerous than the external attacks. These attacks are caused by valid or legitimate users of Clouds. They have all the access to the system and thus can easily bypass the security mechanisms. Here compromised or malicious devices or VMs are actually the legitimate or authorized users of the Clouds. The attackers can gain access to the services of Cloud in a normal manner. They use the valid identity provided by the compromised devices to conceal their malicious behaviors. Therefore, internal attacks generated by the malicious insider devices need more attention.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 654–670, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-052

The insider devices after gaining access to the network can misuse it in the following manner (Boppana and Su, 2007; Essays, 2013):

A. *Packet Dropping*: A malicious device or any other device can drop the packets. Packet dropping is done with either the intention to launch the attack or to save its resources. The attackers successfully drop the packets without being getting noticed. It may also gain the services of other devices for forwarding its own packets.

B. *Device Isolation*: Device isolation refers to preventing the device from communicating with any other device. It is done by an internal malicious legitimate device.

C. *Route Disruption*: An internal attacker device gets it self-added between two endpoints of the communication channel and causes disruption in the communication process.

D. *Modification Based Attacks*: The simplest method to disturb the operations of a network is to perform the attacks, based on modifications. In this type of attacks, the malicious or compromised device announces itself as a better route option. These attacks are performed by modifying the control message fields or altering the route metric value. Some of these attacks are:

- Hop Count Alteration: A compromised malicious device will use a zero value of hop count and reflect itself as a device with the smallest hop count. This, in turn, shows this internally compromised device as a better option.
- Route Sequence Number Alteration: The selection of an optimum path in a network depends on the metric values like hop count, and delay. In general, the lesser the metric value is, better the path option is. This attack is done by altering this value with the lesser value than the previous better value. It is called SYBIL attack.
- Changing Route Information: Change of route information leads to the DoS attacks. For example, a client (A) wants to exchange data with the data center (DC). At client or source side the path header would be A -> R1 -> R2 -> R3 ->R4->R5->DC. If R2 is a compromised routing device then it can change the routing detail to A->R1->R2->R3. As there is no path from R2 to R3, the packets are dropped by R3. As a result, client (A) will not able to access the services from R3. This results in DoS attacks.
- Impersonation Attacks (Spoofing Attacks): In this type of attacks the malicious device use the valid ID of some other device, and the malicious device hides its own MAC or IP address. In the routing protocols, which do not use the source authentication, a malicious device is capable to launch a number of spoofing-based attacks.

E. Attacks Based on Fabrication

- Poisoning of Routing table: Route information of a network is stored in the routing tables. The malicious devices make false entries in the tables which are called poisoning of the routing table. The fake entries are generated by sending a fabricated traffic signal or altering the valid messages from other devices. This may result into creation of routing loops and bottlenecks, selection of non-optimal routes.
- Poisoning of Route Cache: This is a passive attack which occurs usually in dynamic source routing protocol due to a promiscuous mode of updating of the routing table. This type of attack occurs when valid data stored in the routing table is altered, deleted or injected with false information. A device overhearing any packet may make an entry of packet's header to its own route cache, even if that device is not in the path of source i.e. client to the destination. This system is

vulnerable to the attacks by broadcasting a message with a spoofed IP address of other devices. When the devices will receive this message, they will add this new route to their cache and in future, they will communicate using this route to reach the malicious device.

- Fabricating Route Error Messages: This attack occurs when a valid connecting device moves, then the nearby devices broadcast an error message to all the other devices for informing them the inaccessibility of the route for future. The malicious insider device can very easily cause a Denial of Service (DoS) (Wang, 2018) attack by spoofing any node and sending error messages to all the other connecting devices.
- Eavesdropping: The aim of this attack is to gain access to the secret data which is to be protected while two devices are communicating. Data such as private-public key and location are critical from the view of security. Thus, they should be protected from unauthorized access.

4. Data Integrity Techniques for Clouds

The data stored in Clouds could get damaged during the transmission to or from Cloud data storage. Data integrity is to keep data protected from unauthorized modifications. If, there are any modifications in data then, it should be detected. Integrity should be managed at computation as well as the data level. Computation integrity is to keep the program intact during the execution i.e. it should behave like the expected behavior and no malware, insider or malicious user can alter the program which may result into invalid results (Mahmoud et al., 2005; Abbdal et al., 2014).

Cloud computing environment gives the ease to access the data anywhere and anytime. Thus to maintain the data integrity and reduce the threats and risk following techniques are used (Al-Saiyd and Sail, 2013; Aldossary and Allen, 2016):

A. *Secure Hash Function:* The use of a secret symmetric key (k) between the two connecting devices can effectively generate and validate a message authentication $h_k()$. This is done by using one-way cryptographic hash function (h).One-way mathematical function is used for checking data integrity. This function takes a data stream as input and output are reduced to fixed size data. The result of this function can be considered as the fingerprint of the data and is called digest. It is used to maintain integrity in electronic communication. This technique is not affective to protect destination from an attack, when a third party intercepts the sent data and modifies or replaces it and digest of the new message. When the client and server share same private or secret key then, Hash-based Message Authentication Code (HMAC) is generated using a secure hash function. The data file and the resulting hash value is stored in the Cloud. Data file and its corresponding HMAC are provided to the user when a request is made by the user. HMAC can be regenerated for protecting the changes. In this method, the third party is not able to generate the HMAC as they do not have the secret key. This method can be integrated with methods like SHA-1 and MD5.

B. *Digital Signature:* Digital signatures are used to verify data integrity. If after applying the signatures some alterations are made in the original data then a varied digest will be generated. Thus, if the message is not integrated then the validation check will not succeed. This method is also applicable for system or application authentication to

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 654–670, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-052

prevent the remote access of the data. It also restricts the physical access to the prohibited area and provides protection to network management from masqueraders.

C. *Security for Infrastructures:* A different range of cryptographic security techniques are required to provide security services to the sender and the receiver. To accomplish the confidentiality, symmetric encryption keys are distributed. This can be effectively done by the use of public key based key management with a trusted third party (TTP). Interlinked TTPs are needed to provide security services outside the boundaries of an organization. This solution is a comprehensive one. The sender and receiver are able to authenticate each other's identity only if they have each other's public keys. They must depend on a trusted third party to distribute the public keys and authenticate the identity of the party associated with the corresponding key pair.

5. Dynamic Source Routing

A dynamic source routing protocol is a simple and high efficient routing protocol. It is a highly adaptable protocol to the dynamically occurring changes in the network topologies. This is due to the capability of obtaining redundant route information and quick reaction services. The shortcomings of the Dynamic Source Routing are overcome by protocols like SRP and Aridane. The former protocol is capable of providing security only at client and destination. It does not give any mechanism for intermediate devices. The latter protocol overcomes the shortcoming of SRP by using the peer-peer to authentication method (Kuhn et al., 2001; Ashraf and Sujith, 2013).

In Dynamic Source Routing packets are forwarded using the source routing method. The client or source knows the complete path before it begins the transmission. It has two main phases, namely Route Discovery and Route Maintenance. The former phase finds the best of data transmission between the client and server while the latter phase ensures that the communication path is optimized and loop-free. These conditions remain withheld even if the route is changed during the transmission. Route discovery is done by broadcasting a Route Request message (RREQ). The message header of the RREQ contains the sequence of the connecting devices it passes through. A connecting device in the network will not broadcast the same RREQ it has already sent. On receiving the RREQ message the intended receiver or destination will reply by sending a Route Reply (RREP) to client or source. The path information which the destination gets from RREQ packet will be carried by the RREP packet. RREP is generated only when the generated message reaches the intended destination. This RREP travels back to the client or source. This time all the intermediate devices know the route to the client. The failure of the route is detected by the message transmission failure. On receiving the error messages the source and the intermediate devices will remove the path which uses broken link from their path cache.

6. ARCN (Authenticated Routing Protocol for Cloud Networks)

ARCN uses the public key cryptography to mitigate the various security attacks like malicious insider attacks, Spoofing, Falsified routes, DoS etc. ARCN is capable to detect and protect from malicious actions by third parties and peers. This protocol prevents the internal as well as external attacks by using the cryptographic certificates. ARCN uses asymmetric key cryptography. It needs trusted certification. The certificate accommodates the IP address of the connecting device¸ its public key and a timestamp of when the certificate was created and time at which the certificate expires along with the signature by the certification authority. ARCN uses two-step processes: a) certification process and b) route instantiation process to provide a guarantee for

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 654–670, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-052

end-to-end authentication. ARCN depends upon the cryptographic certificates. These certificates are used in the process of route discovery in order to attain authentication, message integrity, and non-repudiation. ARCN requires a trusted authentication server or data center (DC). Only the valid devices or users have the public key of DC. A Device which does not have the valid certificate is considered to be a malicious one. The devices or users use these certificates to prove their identity during the routing of messages (Oberoi et al., n.d.)

7. Introduction of Authenticated Dynamic Routing in Cloud Networks (ADRCN)

Authenticated Dynamic Routing in Cloud Networks (ADRCN) is on-demand secure routing framework. It provides protection against capricious dynamic attackers. It uses symmetric cryptography. It has the capability to prevent the tampering with uncompromised routes and to prevent the attackers. The security of this framework depends on the secrecy and authenticity of keys which are kept at the connecting devices. Authenticated Dynamic Routing in Cloud Networks (ADRCN) ensures point-to-point authentication of the path using a shared key and MAC between the two parties. Authenticated Dynamic Routing in Cloud Networks (ADRCN) assures that 1) the destination of the route discovery process is capable of verifying the initiator, 2) the initiator is capable to verify the intermediate devices, and 3) no intermediate device can make any alteration to the device list in Client Route Request (CRREQ) or Client Route Reply (CRREP) message. ADRCN uses two-step processes for its working: i) Route discovery, and ii) Route maintenance.

Message Authentication Code (MAC) is used to check the accuracy of the message. This is done by the structuring of code using a hash function. The hash function is used on the ID of the sender device and the previous hash message. This makes the receiver sure that the received message is authenticated one. In this framework, each device which receives the CRREQ authenticates itself as well as the message. This is done using the MAC code.

ADRCN is an on-demand secure routing framework based on dynamic source routing. It is capable to combat the compromised routing devices and depends on the symmetric cryptography. ADRCN assures that the destination of a route discovery process is capable to authenticate the client which has generated the request. The client authenticates every intermediate device which lies between the paths to the destination (DC) which is present in the CRREP message. None of the connecting devices can alter the device list in the CRREQ or CRREP messages. The different nomenclature used in this article has been presented in Table 2, which is utilized from equation 1 to 10.

Table 2. Table of notations

A	Client/ Source
DC	Data Center
R_1, R_2, R_3	Routing Devices
p_A	Plain text sent by client A
m_A	Hash value computed over p_A
$K_{A\,DC}$	Shared key of client A and DC
h_{R1}, h_{R2}, h_{R3}	Hash value (h_{R1} calculated over an intermediate device R1 and m_A)
CRREQ	Client Route REQuest
HMAC	Hash MAC code
CRREP	Client Route REPly

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 654–670, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-052

Features

(i) ADRCN uses point to point authentication of a routing message using MAC (Message authentication code) and shared key between the source and destination to check the authentication.
(ii) It doesn't take into account the selfish nodes.
(iii) In order to restrict an attacker from deleting a device from the device list per hop hashing algorithm is used.

Strengths

(i) ADRCN is capable to handle the malicious nodes, which alter and fabricate routing information.
(ii) It is capable to mitigate the cache poisoning attack.
(iii) It is capable to detect and prevent the insider attacks.

The security aim of the protocol is to restrict the attacker to tamper the data using routes, which are uncompromised. ADRCN has the property that no device in between the source and destination can eliminate a device (which is already added in device list) from the device list of path request or reply. In other words, ADRCN is capable to prevent a malicious device from eliminating a device from the device list.

When the client (A) wants to send a message to the destination (DC), then firstly it looks into its local table for an existing path. If no entry is found then only route discovery is done. Discovery of a path is done in two steps:

A. The client or source broadcasts(using limited flooding in this case) the request of path determination to all neighboring devices, and
B. The destination or target i.e. data center sends back the path reply back to the client after it gets the request.

7.1 Client Route Request Broadcast (CRREQ)

The client creates a client route request (CRREQ) and broadcasts it to all its neighboring devices. CRREQ consists of:

$$CRREQ : \{crreq, \text{source}, destination, hash\ , devicelist, MAClist\} \quad (1)$$

Hash is the value, which is determined by per hop hashing function h(x). h(x) gives assurance that no intermediate device or valid device is removed by the malicious device or it modifies the device list order. For the hash function, the first element is determined as $MAC_{S,D}$ (plain text to be sent by source). Device and MAC list store the addresses and MAC values of the intermediate devices. These do not have any initial value. The authenticity of the route request message is verified by all the intermediate devices. Also, the effectiveness of the key is verified.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 654–670, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-052

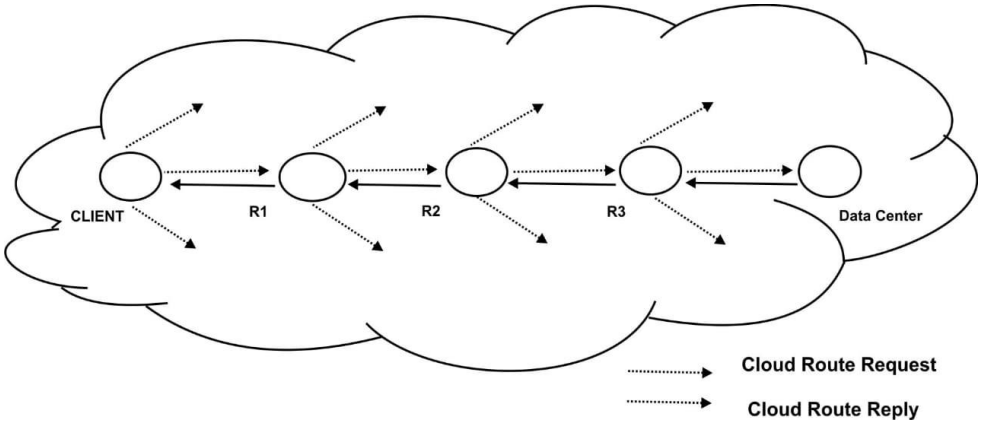

Figure 2. Broadcast of route request and returning of route reply

In Figure 2, client A broadcasts (using limited flooding) path request to all the connected devices in the complete network. Initially, client A broadcasts CRREQ to all its neighbors.

$$\text{A: } \mathrm{p_A} = (\ \mathrm{CRREQ}\ , \mathrm{A}, \mathrm{DC}, [\ ], [\ ], [\ ]\ ), m_A = HMAC_{K_{ADC}}\ (p_A)$$

$$A \rightarrow *\text{: } (\mathrm{p_A}\ , m_A) \tag{2}$$

Device list and MAC list are empty at the initial stage. HMAC value is computed over the plain text (p_A) with a key shared by the client and destination (K_{ADC}). Every intermediate device computes this MAC hash function iteratively along with its own identifier using the publically known one-way hash function (MD5 or SHA-1). For broadcasting the CRREQ limited flooding is used. Every intermediate device which receives the CRREQ first time re-calculates the hash value, adds its identifier to list of identifiers in CRREQ and computes a MAC on the updated CRREQ using a key which is being shared with the destination. This newly computed MAC value is appended to the CRREQ and it is rebroadcasted.

$$R1\text{: } h_{R1} = H(R1, m_A),\ \ p_{R1} = (CRREQ, A, DC, [R1], h_{R1}, [\ ])\ , m_{R1} = HMAC_{K_{R1}}\ (p_{R1})$$
$$R1 \rightarrow *\text{: } \{\mathrm{p_{R1}}, \mathrm{m_{R1}}\} \tag{3}$$

$$R2\text{: } h_{R2} = H(R2, h_{R1}), p_{R2} = (CRREQ, A, DC, [R1, R2], h_{R2}, [m_{R1}]),\ m_{R2} = HMAC_{K_{R2}}\ (p_{R2})$$
$$R2 \rightarrow *\text{: } \{\mathrm{p_{R2}}, \mathrm{m_{R2}}\} \tag{4}$$

$$R3\text{: } h_{R3} = H(R3, h_{R2}), p_{R3} = (CRREQ, A, DC, [R1, R2, R3], h_{R3}, [m_{R1}, m_{R2}]),$$
$$m_{R3} = HMAC_{K_{R3}}\ (p_{R3})$$
$$R3 \rightarrow *\text{: } \{\mathrm{p_{R3}}, \mathrm{m_{R3}}\} \tag{5}$$

Source routing is adapted by the client (A) to connect to the destination DC, through the intermediate devices R1, R2, and R3. The framework establishes a hash chain at the destination (DC).

$$H(\ R3, (H\ (R2, H(R1\ , HMAC_{K_{ADC}}(A, DC))))$$

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 654–670, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-052

where $HMAC_{K_{ADC}}(A, DC)$ denotes message m_A, HMAC code generated by a shared key between the source (A) and destination (DC). The one-way hash function (H) authenticates the contents in the chain, and $HMAC_{K_{ADC}}(A, DC)$ authenticates the client- data center relation.

7.2 Unicast of Client Route Reply (CRREP)

When data center or the destination receives the path request message, it checks the reliability of the device list. It also checks that the used keys have been released within a specific time or not. If all the above conditions are met then data center (DC) accepts the request of path establishment and sends a path reply back to the client (A) in the opposite direction i.e. from destination to source. It also verifies the per hop hash value by re-computing the MAC value of source as well as of all the intermediate devices. Then verification of the MAC values in the CRREQ is done. The Client Route Reply (CRREP) consists of

$$CRREP : \{crrep\ , destination, source, devicelist, MAClist\} \tag{6}$$

The device list is the obtained in CRREQ and MAC of the destination ($M_{\text{destination source}}$) on all these elements with a key shared by destination and the source is used. The destination i.e. the data center will generate a CRREP as follows:

$$DC: p_{DC} = (CRREP\ , DC, A, [R1, R2, R3], [m_{R1}, m_{R2}\ , m_{R3}\ ])\ , m_{DC} = \text{HMAC}_{\text{K}_{\text{DC C}}}(\text{p}_{\text{DC}})$$
$$DC \rightarrow R3: (\text{p}_{\text{DC}}, m_{DC}\ , [\ ]) \tag{7}$$

The intermediate device R3 on receiving the CRREP will add its own secret key i.e. K_{R3} and will send CCREP towards the source.

$$R3 \rightarrow R2: (\text{p}_{\text{DC}}, m_{DC}\ , [\text{K}_{\text{R3}}]) \tag{8}$$

The same process is repeated by all the intermediate devices between the destination and source.

$$R2 \rightarrow R1: (\text{p}_{\text{DC}}, m_{DC}\ , [\text{K}_{\text{R3}}, \text{K}_{\text{R2}}]) \tag{9}$$

$$R1 \rightarrow A: (\text{p}_{\text{DC}}, m_{DC}\ , [\text{K}_{\text{R3}}, \text{K}_{\text{R2}}, \text{K}_{\text{R1}}]) \tag{10}$$

As the client A receives the CRREP it calculates h'. If the h (HMAC computed by the data center) and h'(HMAC computed by the client) are identical then the client accepts the route returned in the CRREP. Hence the path established between the client and data center is authenticated as well as integrity is also maintained as no intermediate device can alter the path to perform the internal attack. The path is authenticated as well as the integrity of the path is maintained as no intermediate device can alter the path.

At the destination, DC can compute m_{A} as the information of p_{A} is contained in p_{R3}. The value of h_{R3} is calculated dynamically by the destination DC. DC uses the list embedded in the CRREQ packet to calculate the h_{R3}'. h'_{R3} is compared with h_{R3} and if the result is same ($\text{h}_{\text{R3}}' = \text{h}_{\text{R3}}$) then data is secure, otherwise some alteration has been done during communication.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 4, No. 3, 654–670, 2019
https://dx.doi.org/10.33889/IJMEMS.2019.4.3-052

8. Simulation and Implementation

The proposed framework is implemented in Matlab. The simulation environment involves 50 routers. ADRCN is capable of preventing and detecting the malicious insider attacks in Clouds. In Figure 3, a route is to be established from the client A to the destination DC. Client route discovery is done by client A through the intermediates devices B, J, I, F, C.

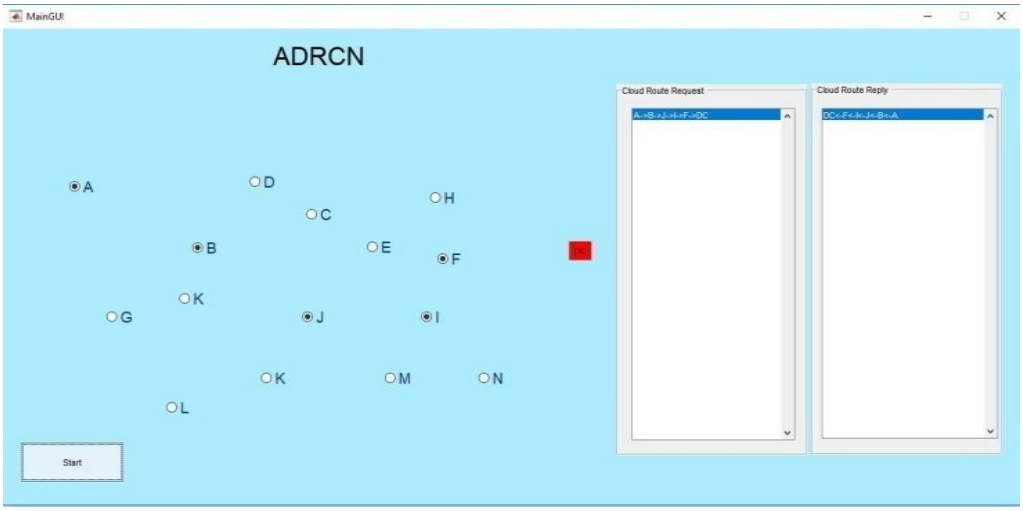

Figure 3. Basic outline of ADRCN implementation

ADRCN assures that the path from the client to DC is authenticated as well as the integrity of the path is maintained as only the valid devices are allowed to participate in the routing process. In the route reply process, every intermediate device appends its own key to CRREP packet. This ensures that the path integrity is maintained. Hash chain value computed by the DC during the request process is compared by the hash value computed by the client during the reply process. If both the values are same then the path is authenticated as well as the integrity is maintained.

Once route discovery is done by the client A and destination DC receives the CRREP packet, then DC sends a route reply through the reverse path i.e. DC, C, F, I, J, B, A. The client A on receiving the route reply will start the communication. If any malicious insider device tries to misuse the path then it will be detected as any change in the routing will lead to the mismatch in the hash values computed by the source and data center.

9. Results

ADRCN framework is implemented for Cloud-based environments in Matlab. When any router or device sends a request for the path establishment then the request is fulfilled according to the ADRCN framework. Initially, a request of path establishment is sent to the destination (CRREQ) and, once the destination receives the request it sends back the route reply (CRREP) to the client. This framework ensures that the path of CRREQ and CRREP are same. No intermediate or malicious device is allowed to interfere in the communication process. Hence the framework is capable enough to mitigate the malicious insider attacks in the IaaS of Cloud-based environment.

As shown in Table 3, ADRCN is better than other technologies as it achieves all the security factors.

Table 3. Comparison of security factors

Parameters	ARCN	ADRCN	Dynamic Routing
Path Authentication	Yes	Yes	No
Data Confidentiality	Yes	Yes	No
Path Integrity	No	Yes	No
Non-Repudiation	Yes	Yes	No

10. Conclusion and Future Work

A Cloud is at risk of numerous types and different approaches of attacks. In this work, the authors have proposed a security routing framework viz. Authenticated Dynamic Routing in Cloud Networks (ADRCN) to detect and prevent the malicious insider attacks in Cloud-based environments. The proposed framework uses symmetric key cryptography to build security against the various types of internal attacks. The client, the receiver and the intermediate devices are authenticated by secret key, hashing, and MAC code. The aim of Authenticated Dynamic Routing in Cloud Networks (ADRCN) is to maintain path authentication and path integrity in the Cloud-based environment. ADRCN is capable of preventing and detecting the malicious insider attacks in Clouds.

As a future perspective, the proposed solutions will be improved to decrease its overhead. Also, more simulations can be done in different scenarios and various Cloud platforms.

Conflict of Interest

The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgement

The authors would like to express their sincere thanks to the anonymous reviewers and there constructive comments for the improvement of this paper. Also, we gratefully acknowledge financial support from D.A.V. Centenary College, Faridabad. We extend our sincere thanks to them for the unconditional support.

References

Abbdal, S. H., Jin, H., Zou, D., & Yassen, A. A. (2014). Secure third party auditor for ensuring data integrity in cloud storage. *Proceedings - 2014 IEEE International Conference on Ubiquitous Intelligence and Computing, 2014 IEEE International Conference on Autonomic and Trusted Computing, 2014 IEEE International Conference on Scalable Computing and Communications and Associated Symposia/Workshops, UIC-ATC-ScalCom 2014*, 510–517. https://doi.org/10.1109/UIC-ATC-ScalCom.2014.17

Aldossary, S., & Allen, W. (2016). Data security, privacy, availabilifty and integrity in cloud computing: issues and current solutions. *International Journal of Advanced Computer Science and Applications*, *7*(4), 485–498.

Al-Saiyd, N. A., & Sail, N. (2013). Data integrity in cloud computing security. *Journal of Theoretical and*

Applied Information Technology, *58*(3), 570–581.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2010). A veiw of cloud computing. *Communications of the ACM*, *53*(4), 50–58.

Ashraf, U., & Sujith, S. (2013). Dynamic source routing. Retrieved from http://www.slideshare.net/ashrafmath/dynamic-source-routing

Bhatt, N., & Anand, A. (2017). Modeling and characterizing software vulnerabilities. *International Journal of Mathematical, Engineering and Management Sciences*, *2*(4), 288–299.

Boppana, R. V, & Su, X. (2007). Secure routing techniques to mitigate insider attacks in wireless ad hoc networks. *IEEE Wireless Hive Networks Symposium*, (2). Retrieved from http://www.cs.utsa.edu/faculty/boppana/papers/Whns07-preprint.pdf

Chandramohan, D., Vengattaraman, T., Rajaguru, D., Baskaran, R., & Dhavachelvan, P. (2013). A novel framework to prevent privacy breach in cloud data storage area service. *2013 International Conference on Green High Performance Computing, ICGHPC 2013*, *2011*(40), 1–4. https://doi.org/10.1109/ICGHPC.2013.6533903.

Chen, Y., Li, L., & Chen, Z. (2018). An approach to verifying data integrity for cloud storage. *Proceedings - 13th International Conference on Computational Intelligence and Security, CIS 2017*, *2018–Janua*, 582–585. https://doi.org/10.1109/CIS.2017.00135.

Dara, S., Gopularam, B. P., Muralidhara, V. N., & Nalini, N. (2016). Experimental evaluation of network telemetry anonymization for cloud based security analysis. *Proceedings - 2015 IEEE International Conference on Cloud Computing in Emerging Markets, CCEM 2015*, 1–7. https://doi.org/10.1109/CCEM.2015.10.

Dewangan, B. K., Agarwal, A., & Venkatadri, M. (2019). Energy-aware autonomic resource scheduling framework for cloud. *International Journal of Mathematical, Engineering and Management Sciences*, *4*(1), 41–55.

Egawa, T., Nishimura, N., & Kourai, K. (2012). Dependable and secure remote management in IaaS clouds. *CloudCom 2012 - Proceedings: 2012 4th IEEE International Conference on Cloud Computing Technology and Science*, 411–418. https://doi.org/DOI: 10.1109/CloudCom.2012.6427597

Essays, U. (2013). Security for insider attacks in mobile ad hoc networks. Retrieved from https://www.ukessays.com/dissertation/examples/information-technology/adding-security-against-insider.php?vref=1

Giri, M. S., & Gaur, B. (2015). A survey on data integrity techniques in cloud computing. *International Journal of Computer Applications*, *122*(2), 975–8887.

Gor, M., & Jain, G. (2016). Survey on cloud database security using onion encryption techniques. *International Institution for Technological Research and Development*, *1*(6). Retrieved from http://www.iit-rd.com/FinalPaper/FinalPaperSurvay on Cloud database security using onion encryption techniques150078.pdf

Hariharasitaraman, S., & Balakannan, S. P. (2018). On fractal way of checking data storage integrity in Cloud storage. *Proceedings of the 2017 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing, INCOS 2017*, *2018–Febru*, 1–5. https://doi.org/10.1109/ITCOSP.2017.8303135.

Jain, A., & Kumar, R. (2014). A taxonomy of cloud computing. *International Journal of Scientific and Research Publications*, *4*(7), 1–5.

Jain, A., & Kumar, R. (2016). Confidentiality enhanced security model for cloud environment. *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies - ICTCS '16*, 1–6. https://doi.org/10.1145/2905055.2905199.

Jones, N., Arye, M., Cesareo, J., & Freedman, M. J. (2011). Hiding amongst the clouds: a proposal for cloud-based onion routing. *USENIX Workshop on Free and Open Communications on the Internet FOCI*. Retrieved from https://sns.cs.princeton.edu/docs/cor-foci11.pdf.

Kalpana, P., & Singaraju, S. (2012). Data security in cloud computing using RSA algorithm. *International Journal of Research in Computer and Communication Technology*, *1*(4), 143–146.

Kaur, J., & Singh, J. (2013). Monitoring data integrity while using TPA in cloud environment. *International Journal of Advanced Research in Computer Engineering & Technology*, *2*(7), 2236–2240.

Kavuri, S. K. S. V. A., Kancherla, G. R., & Bobba, B. R. (2014). Data authentication and integrity verification techniques for trusted/untrusted Cloud servers. *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014*, 2590–2596. https://doi.org/10.1109/ICACCI.2014.6968657.

Khan, I., Rehman, H. U., & Anwar, Z. (2011). Design and deployment of a trusted Eucalyptus Cloud. *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, 380–387. https://doi.org/10.1109/CLOUD.2011.105.

Kopachevsky, I., Kostyuchenko, Y. V, & Stoyka, O. (2016). Land use drivers of population dynamics in tasks of security management and risk assessment. *International Journal of Mathematical, Engineering and Management Sciences*, *1*(1), 18–25.

Kuhn, D. R., Hu, V. C., Polk, W. T., & Chang, S.-J. (2001). NIST special publication 800-32 - introduction to public key technology and the federal PKI infrastructure. *NIST Special Publication*, (February), 1–54. https://doi.org/10.6028/NIST.SP.800-32.

Leena, & Rao, M. A. K. (2012). Centralized database security in cloud. *International Journal of Advanced Research in Computer and Communication Engineering*, *1*(8), 544–549.

Mahmoud, A., Sameh, A., & El-Kassas, S. (2005). Reputed authenticated routing for ad hoc networks protocol (Reputed-ARAN). *2nd IEEE International Conference on Mobile Ad-Hoc and Sensor Systems, MASS 2005*, *2005*, 787–794. https://doi.org/10.1109 /MAHSS.2005.1542872.

Nafi, K. W., Kar, T. S., Hoque, S. A., & Hashem, M. M. A. (2012). A newer user authentication , file encryption and distributed server based cloud computing security architecture. *International Journal of Advanced Computer Science and Applications*, *3*(10), 181–186.

Oberoi, P., & Mittal, S. (2017). Survey of various security attacks in clouds based environments. *International Journal of Advanced Research in Computer Science*, *8*(976), 405–410.

Oberoi, P., & Mittal, S. (2018). *Review of CIDS and techniques of detection of malicious insiders in cloud-based environment*. *Advances in Intelligent Systems and Computing* (Vol. 729). https://doi.org/10.1007/978-981-10-8536-9_11.

Oberoi, P., Mittal, S., & Kumar, R. (n.d.). ARCN: Authenticated routing on cloud network to mitigate insider attacks on IAAS , unpublished.

Patil, S., & Rai, N. (2018). An efficient data integrity & data recovery with two TPAs in cloud data storage. *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing, ICECDS 2017*, 1301–1304. https://doi.org/10.1109/ ICECDS .2017.8389654.

Shen, J., Liu, D., He, D., Huang, X., & Xiang, Y. (2017). Algebraic signatures-based data integrity auditing for efficient data dynamics in cloud computing. *IEEE Transactions on Sustainable Computing*, *3782*(c), 1–1.

Sirisha, A., & Hiranmayee, N. (2015). Data integrity check and efficient data storage in cloud using hashfunctions, blowfish and RSA. *International Journal of Advanced Research in Computer Science and Software Engineering*, *5*(4), 513–518.

Tamura, Y. (2017). Dependability analysis tool based on multi-dimensional stochastic noisy model for cloud computing with big data. *International Journal of Mathematical, Engineering and Management Sciences*, *2*(4), 273–287.

Wang, L. (2018). Big data and IT network data visualization. *International Journal of Mathematical, Engineering and Management Sciences*, *3*(1), 9–16.