International Journal of Research in Advent Technology, Vol.7, No.8, August 2019
E-ISSN: 2321-9637
Available online at www.ijrat.org

Querying the Data Provenance from Ethereum Blockchain

Navya Gouru , Nagalakshmi Vadlamani

Abstract— The potentiality of Blockchain technology is widespread and applied to diverse fields. Blockchain is a distributed ledger of transactions that store immutable records in chronological order in an append-only mode. Hence, humongous data is stored on the blockchain and will continuously expand over time. Blockchain has been rapidly adopted by many businesses for storing the provenance data because of its salient features like immutability, robustness and tamperproof. Blockchain stores data provenance as transactions that are collected from sources like a centralized cloud or decentralized cloud that helps in identifying cybercrimes. This paper emphasizes on the different approaches of querying the data provenance transactions stored in Ethereum Blockchain based on various search parameters using REST API web services. The approach not only queries based on the first-class data elements like blocks, transactions, account address and contract address but also queries based on the provenance data stored on the Ethereum Blockchain explained with a use case LegalProv.

Keywords- Blockchain; Query, Centralized Cloud; Decentralized Cloud; Data Provenance;

I. INTRODUCTION

As the name suggests, Blockchain is a chain of blocks where a set of transactions are grouped into blocks and all the blocks are linked to each other cryptographically. A transaction could be an exchange of cryptocurrency or could be a simple data that is broadcasted to all the nodes in the peer-to-peer network. Each peer in the network holds an entire copy of the Blockchain. Each transaction is validated and combined to form a block and a new block is appended to the existing Blockchain. Ethereum blockchain is a Turing complete language that allows developing smart contracts which are later compiled into bytecode and executed by EVM (Ethereum Virtual Machine). Ethereum has its own cryptocurrency called Ethers that are used as a payment source for processing services on the Ethereum network. Blockchain technology has gained popularity with its best-known implementation Bitcoin Nakamoto, S. [1]. There are many use cases that store provenance data in blockchain for maintaining secure tamper-proof provenance. Data provenance using blockchain-based helps in traceability for agricultural products J. Hua et al. [2] in protecting food safety to solve the trust issue among different stakeholders for the supply chain in agricultural products. Tracking the provenance data in mining and metal industries S. Mann et al. [3] which is a crucial unit of the global economy which abates the issues like traceability and increase transparency among various suppliers. IoT also Implements blockchain-based data provenance U. Javaid et al. [4] for integrity by using Physical Unclonable Functions (PUFs) and ethereum blockchain smart contracts to tolerate tampering attacks. Multimedia also utilizes blockchain-based data provenance for privacy protection A. Vishwa et al. [5] and there are various other sectors like financial institutions, decentralized government, health care, real estate, IoT and also cloud computing that uses blockchain technology. Blockchain helps in cloud security by tracking the data provenance of operations performed on the centralized cloud Ramachandran et al. [6] and decentralized cloud Navya et. al. [7] Now storing such provenance data on the blockchain is more efficient because of its robust features. Ethereum blockchain stores humongous data as transactions and efficient approaches are required in retrieving the data from the blockchain. The major components in Ethereum blockchain for data retrieval can be done based on Block number, Transaction hash, EOA (Externally owned accounts) and CA (Contract address) which are considered as first-class elements.

In this paper, the authors emphasize on various approaches to querying the data provenance from ethereum blockchain that includes using web3JS, using Rest API's web services to fetch from MongoDB and also SQL.

The other sections of this paper are formulated as follows. Section 2 discusses the background and related work of the ethereum blockchain with first-class elements. Various approaches to querying the ethereum blockchain are discussed in explained in section 3. Section 4 explains about querying the ethereum blockchain with search parameters of provenance data stored as input data in a transaction with REStful API's. Finally, section 5 concludes the paper.

II. BACKGROUND AND RELATED WORK

Efficient data retrieval approaches are mandated to any storage platform as data stored serves no purpose without retrieval. The transactions once stored in Blockchain cannot be edited or deleted hence data will only grow over time. This leads to massive data storage in blockchain and retrieving the required information from this humongous data becomes quite tedious.

Ethereum blockchain explorer retrieves the blockchain data with the first-class data elements like blocks, transactions and accounts. The most well-known ethereum blockchain

Manuscript revised on August 10, 2019 and published on September 10, 2019
Navya Gouru , *GITAM Institute of Science, GITAM (Deemed to be University), Visakhapatnam, India. Email: navya.phd@gmail.com*
Nagalakshmi Vadlamani, *GITAM Institute of Science, GITAM (Deemed to be University), Visakhapatnam, India.*

doi: 10.32622/ijrat. 77201913

explorers are etherscan, etherscan explorer, [8], ethplorer, ethplorer explorer, [9], etherchain, explorer [10].

As stated, Blockchain is a collection of blocks that hold transactions that are executed between two accounts by invoking a smart contract. Here, the first-class elements in Ethereum blockchain are accounts, transactions, blocks and smart contracts. All these first-class elements unique identities are used to query the Ethereum blockchain.
The structure of a block is classified into two types called Block Header and the set of Transactions that are mined in the block. Each block holds the hashing information about the previous to form a link and there exists a Merkle tree of hashes for state root, transaction root and receipt root and other fields. The key point is storing the data in the blockchain and the entire concept, the structure of the blockchain goal is to maintain tamper-proof, immutable storage of the data. The data enters the blockchain by interacting with a smart contract with an account address (EOA). Internally, the smart contract creates logs by triggered events. Events are triggered either by an account address or by another contract address. Now, retrieving the data from ethereum blockchain in tedious.

Accounts: Ethereum accounts are broadly classified into two types known as Externally Owned Accounts (EOA) and Contract Accounts (CA) which have an address of size 20 bytes each. Externally owned accounts are accounts that are controlled by a user with a private key. EOA is used to sign the transaction with a private key to get broadcasted into the network. As an example, to send Ethers to another user both the users should hold an EOA account where the sender of the accounts signs the transaction with his private key. On the other side, contract accounts are created when deploying a smart contract and is controlled by code in the smart contract. The Contract address can be invoked by another EOA or CA.
The components of the Account are:

- Nonce: Number of transactions or contracts created
- Balance: Number of Wei's in ethers
- storageRoot" Root node hash in Merkle tree
- codeHash: Code hash of EVM or an empty string

Transaction: Transaction is a manner of interactions with the ethereum blockchain network that modifies the current state of the network. The transaction occurs between different types of accounts. Transactions are broadly classified into two types namely, messages call and contract creations where both are initiated by an externally owned account and broadcast to the network. Contract creation is a transaction created for deploying a new contract whereas message calls are transactions that are created when interacting with existing contracts.
The attributes of a transaction are as below:

- Transaction Hash: unique identifier generated when a transaction is created.
- Status: status of the transaction.
- Block: number of the block in which the transaction is mined.
- Timestamp: date and time at which the transaction is mined.
- From: account address of the sender.
- To: account address of the receiver.
- Value: the value being transacted in Ethers.
- Transaction Fee: amount paid to the miner in Ethers.
- Gas Limit: the highest amount of gas provided for the transaction.
- Gas Used by Transaction: the amount of gas utilized by this transaction.
- Gas Price: cost per unit of gas for the transaction.
- Nonce: index of the transaction, starts at 0.
- Input Data: additional information for this transaction.

Blocks:
A blockchain is a list of blocks where each block holds a group of transactions. A typical block consists basically of two types of information known as Block Header and a collection of Transactions.
The attributes of the block are:

- Block number: unique integer identifier of a block.
- Block height: refers to the length of the blockchain.
- Timestamp: data and time at the block's inception.
- Transactions: number of transactions in the block.
- Mined by: account address of the miner of the block.
- Block reward: reward in ethers for mining the block.
- Uncles reward: an extra reward for including uncles as part of the block.
- Difficulty: the amount of effort to mine the block adjusts according to time.
- Total difficulty: the total difficulty of the chain until this block.
- Size: determined by block's gas limit.
- The gas used: total gas used in the block and its percentage.
- Gas limit: provided by all the transactions in the block.
- Extra data: extra information included by the miner.
- Hash: hash of previous block header.
- Parent hash: hash of the parent block.
- Sha3uncles: ethereum JS RLP encoding technique.
- Nonce: value used to validate proof of work.

EtherQL F. A. Pratama et. Al. [11] a query layer for Ethereum provides efficient queries to retrieve data from ethereum blockchain such as range queries and top-k queries. EtherQL is designed to also aid data analysts and researchers by providing different levels of abstraction. To achieve this, the authors designed a middleware that synchronizes data from the Ethereum public network with a built-in client in real-time and presents the users with a query layer for efficient access to the blockchain data. This layer has four modules: sync manager, handler chain, persistence framework and developer interface. The design used MongoDB, a NoSQL database, as the external storage to save the output from the handler chain. The data stored in MongoDB can be fetched using various queries in an API interface. The authors of M. Muzamma et. Al. [12] extended the functionality of EtherQL by including Retrieval Queries with multiple search parameters, Aggregate Function Queries

doi: 10.32622/ijrat. 77201913

like count, sum, avg, min, max and Ranking query support for sorting the data. ChainSQL S. Bragagnolo et.al. [13] In this study, the ChainSQL and its unique functions with the help of three use cases by implanting them between the client application and the database as middleware. The first case is a multi-active, tamper-proof database, the second case is a recovery system from a data-level disaster and the third case is an audit middleware. The efficiency of the design is shown with a thorough experimental study. ChainSQL is the first design to present blockchain's tamper-proof and rapid query processing features in distributed databases. The usefulness of ChainSQL can be seen from its use in the areas of finance and supply chain.

III. APPROACHES IN QUERYING ETHEREUM BLOCKCHAIN

Broadly, there are two approaches in querying the ethereum blockchain based on the search parameters.

1. To directly query the blockchain using web3 javascript implementation with web3JS commands that interact between web applications and ethereum blockchain. Using web3js commands querying the blockchain can be done on the first-class elements like account address, contract address, transaction hash and block.

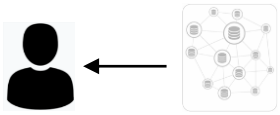

Figure 1: Retrieving directly from Blockchain

2. To maintain a local copy of the blockchain database and query the records from the local database instead of the ethereum blockchain.

Figure 2: Retrieving from local database

The event listener listens on the ethereum network for a new block and automatically detects and captures the block it is generated. In the ethereum network, it usually takes 14-30 seconds for the generation of a new block. The data from the block is extracted and stored in the local MongoDB database in BSON (Binary JSON) format. BSON is the binary encoding of JSON documents used by MongoDB to store data in collections. The data from the database is retrieved by the user in JSON format by using the REST API.

Since the majority of the developers are comfortable by querying with SQL hence maintaining a local ethereum database provides ease to the developer and this includes local databases like MongoDB, Postgres, LevelDB, SQL, etc. There are two popular choices to query eth events [15] namely, elastic search query DSL and ethereum SQL with Postgres database.

i) Accessing Eth Events using the Ethereum SQL Database: Eth events generate a free account to query ethereum blockchain with Postgres PGAdmin4. The benefits provided by eth events using ethereum SQL are:

- fetching the data quickly in real-time with the recent blocks mined.
- provides better flexibility with regards to searching the queries.
- Provides a virtualization layer.

The process to load the data from the ethereum blockchain to the local database which is SQL or NoSQL is like transferring blockchain content which is on-line transaction processing (OLTP) to on-line analytical processing (OLAP) that provide feasibility to read the data from a local database.

ii) Eth Events using Elastic Search: Elastic search provides high-speed data access for both simple and complex queries stored in ethereum. Elasticsearch supports Domain-specific Language to query ethereum blockchain-based on JSON.

Helpful Hints

IV. QUERYING ETHEREUM PROVENANCE DATA

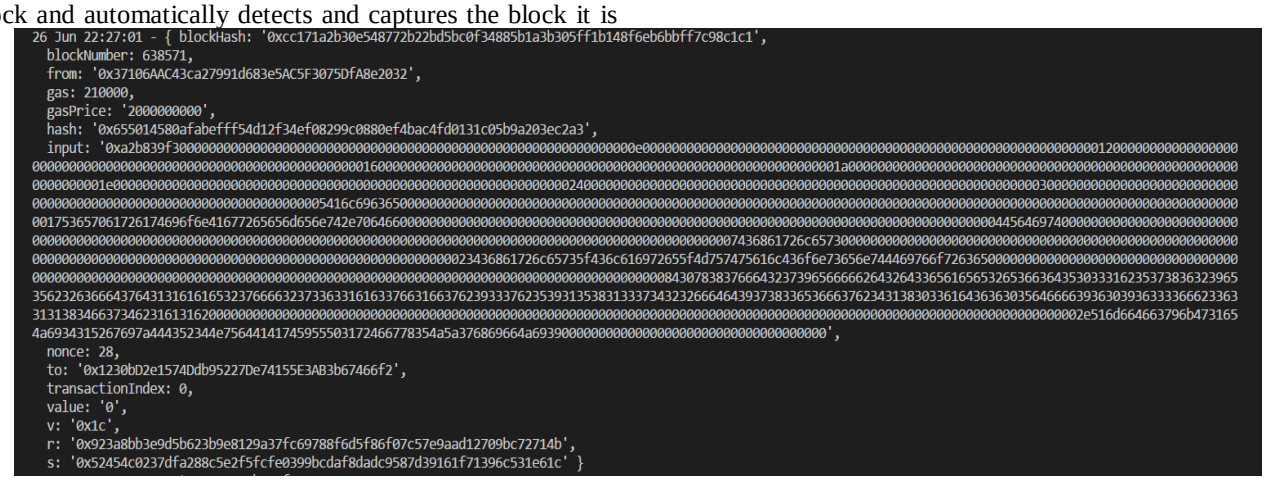

Figure 3: Transaction Details

doi: 10.32622/ijrat. 77201913

When a transaction is mined and confirmed in Ethereum blockchain network each transaction has a unique i)transaction hash and an ii) account address that is the from address which has signed and initiated the transaction iii) contract address which is executed to commit the transaction and lastly, iv) each transaction is stored in a block. Here all the first-class elements like transaction, account address, contract address and block have unique identifiers that are used as parameters to query from the Ethereum blockchain.

Consider a simple use case, called LegalProv which collects data provenance of legal matters during the transfer of sensitive Documents between an Attorney and a Client associated with a Matter that is stored in decentralized cloud IPFS and securely tracks data provenance using Blockchain.

Consider a scenario where an Attorney *Alice* is sending client *Charles* a document with name *SeperationAgreement.pdf* by uploading to IPFS with the Matter name as *"Charles_Claire_MutualConsentDivorce"* and IPFS hashkey and Digital Signature.

LegalProv is storing the provenance data for the above said scenario in blockchain as a transaction which is generated as shown in figure 3. From the figure 3, input data holds the provenance data in an encoded form. The decoded input provenance data is:

```
[ 'Alice'"uploaded",
"Document Name:" 'SeparationAgreement.pdf',
"given permission:" 'Edit',
"to:" 'Charles',
"MatterName:"
'Charles_Claire_MutualConsentDivorce',
"Digital Signature:"
'0x5647fe0c4219cfdfaa53f5319986fe8a80f24278c0d6105
6643e1ee93b6f8c155a0a17440d9e94838fbb42acab32681fb
1fd1821c554b93ccc1dec5ada80e5431b',
"IPFSHasKey":'QmfFcykG1eJi41RgizDCR4NudAAtYUP1rFgx
5JZ7hifJi9',
"EncSharedKey:"'eJwVj01OhTEIRffSsQNaKC1uwYlb4KfEL/
ElxjyNxrh3cUbCPZfDT7s+22PrHqYDaCIRC3Tfo29zxMUYsLQ9
tPP2cm7PH/Z6+dP5LgZoH5MAUhIesEgHK/sO9bTwKtEzYY+tXb
cCRi4y6wnIdmSpI8w9VYdMU3PqNmFZYcIT8rAm1wGG3JMgcouh
owkyjIxw8V5WfpXW+/183cvoBJgVjjOk66CZTInrCAzV2lT+pl
7BcNs9D8B2qarFqYEutnb2Nf8Fc6Uvo3nqeWTBGsC74qJhTO33
D/rEWFY=","Document Hash:"
'phsbekhjKOp7Ifo7NMF2At8LkIX5NiIYQbadCh0i72A=]
```

The above-decoded information is stored in private ethereum blockchain built dedicatedly for LegalProv.

If a user wants to retrieve the provenance data by giving the search parameters like his account address and other search parameters like Document Name, Permission, Matter Name, Digital Signature and IPFS hashkey which are stored in blockchain input data of a transaction initially the first-class elements should be queried.

i) Retrieve ethereum blockchain transactions based on contract address:

Ethereum network has massive data that has many smart contracts deployed. Form the entire network of transactions first query the transactions that are specific to the contract address used in LegalProv. Web3JS Web3.JS API, [14] a javascript implementation used to interact between a web application and ethereum blockchain by using getPastLogs.

```
web3js.eth.getPastLogs({fromBlock: '0x0', to:
'MaxBlockNumber',address:contractAddress>});
```

getPastLogs consists of transaction hashes for each transaction that initiated on this network with a specific contract address used for LegalProv. The getPastLogs, sequentially access the records between the specified block numbers and identifies the transactions with the mentioned contract address provided as a search parameter.

ii) Retrieve ethereum blockchain transactions based on transactions hash:

Based on each transaction hash from pastLogs, transaction details can be retrieved with getTransaction which holds first-class element identifiers like blockNumber, from the address which is an address account, to address which is a contract address and the input field which actually stores the provenance data as a transaction. This can be achieved using below web3 command:

```
web3js.eth.getTransaction(transactionHash)
```

iii) Retrieve ethereum blockchain transactions based on transactions hash:

Based on the blockNumber from the getTransaction the block details can be retrieved from blockchain

```
web3js.eth.getBlock(blockNumber);
```

In all the cases of retrieving the first-class data objects, the search parameters are for Block, Transaction, from or to address, contract address that is either a hash or an address. User should always track the transaction hashes to retrieve provenance data stored in that transaction.

All the first-class data objects are basically to retrieve the data stored in the blockchain. The *input* field in the transaction as in Figure 3 is the encoded information of the provenance data and should be decoded with ABI decoder which give the decoded data in key-value pairs of name, value and type as shown in the example below:

```
params:
{[{ name: '_consignor', value: 'Alice', type: 'string' },
{name:'_documentName',value:'SeparationAgreement.pdf',
type: 'string' },
{ name: '_permission', value: 'Edit', type: 'string' },
{ name: '_consignee', value: 'Charles', type: 'string' },
{ name: '_matterName', value: 'Charles Family Case', type:
'string' },{name: '_digitalSignature', value:
'0x87fd279effbd2d3eaee2e6cd5031b578629e5b2cfd7d11aae2
7ff273c1ac7f1f7b937b59158137422fdd9783e6f7b41803ad660
5dff9609633fb3c1184f74b1a1b',type:'string'},{name:'_ipfsHas
h',value:'QmQX5J1H7ak8SR85nrUaUgoY7ZLCxQrw8FK6N
hVffyyY3s',type: 'string',}
{name:'_encsharedkey',value:'eJwVj01OhTEIRffSsQNaKC1u
wYlb4KfEL/ElxjyNxrh3cUbCPZfDT7s+22PrHqYDaCIRC3T
fo29zxMUYsLQ9tPP2cm7PH/Z6+dP5LgZoH5MAUhIesEgH
K/sO9bTwKtEzYY+tXbcCRi4y6wnIdmSpI8w9VYdMU3Pq
NmFZYcIT8rAm1wGG3JMgcouhowkyjIxw8V5WfpXW+/18
3cvoBJgVjjOk66CZTInrCAzV2lT+pl7BcNs9D8B2qarFqYEu
tnb2Nf8Fc6Uvo3nqeWTBGsC74qJhTO33D/rEWFY=',type:'st
ring'},{name:'_hashvalue',value:'phsbekhjKOp7Ifo7NMF2At8
LkIX5NiIYQbadCh0i72A', type: 'string' }] }
```


doi: 10.32622/ijrat. 77201913

The API endpoint allows users to retrieve the provenance data from ethereum blockchain using the decoded key-value pairs of the input data. /api/select/account/provenancedata/:filtertype that Retrieves exclusive data Provenance records based on the filter type given document name

When a user provides search parameters like documentName, permission, matter name, digitalSignature, ipfsHash, encsharedkey, hashvalue to retrieve records which are signed by him and the endpoints are specified to the search parameters that retrieve the records. Lastly, depending on the parameters provided by the user to retrieve the transaction the specific key values of the searched parameters will be retrieved by using the REST API's web services. Finally, the records with the contract address used by the application and account address of the user and provenance data parameters, the transactions are retrieved from ethereum blockchain.

V. CONCLUSION

Blockchain has been adopted by many organizations and hence massive amount of data is stored in the blockchain. Since blockchain is an append-only distributed network accessing the data is a challenging issue. This paper discusses various approaches in retrieving the data directly from blockchain and maintain a local copy of the database either a SQL or NoSQL database and then retrieve the required data based on searched parameters. This paper also explains how REST apis are used to retrieve the data provenance stored as input data in a transaction. Querying the data directly from blockchain helps to retain the robust features of distributed systems and by querying with local databases should surface the threats involved with centralization.

REFERENCES

[1] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf.
[2] J. Hua, X. Wang, M. Kang, H. Wang and F. Wang, "Blockchain Based Provenance for Agricultural Products: A Distributed Platform with Duplicated and Shared Bookkeeping," 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, 2018, pp. 97-101.doi: 10.1109/IVS.2018.8500647. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8500647&isnumber=8500355
[3] S. Mann, V. Potdar, R. S. Gajavilli and A. Chandan, "Blockchain Technology for Supply Chain Traceability, Transparency and Data Provenance", ICBTA, 2018.
[4] U. Javaid, M.N. Aman and B. Sikdar, "BlockPro: Blockchain based Data Provenance and Integrity for Secure IoT Environments", BlockSys, 2018.
[5] A. Vishwa and F. K. Hussain, "A Blockchain based approach for multimedia privacy protection and provenance," 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bangalore, India, 2018, pp. 1941-1945.doi: 10.1109/SSCI.2018.8628636 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8628636&isnumber=8628618
[6] Ramachandran, A., and Kantarcioglu, M., "DataProv: Using Blockchain and smart contracts for secure data provenance management", arXiv:1709.10000v1 [cs.CR], 2017
[7] N. Gouru and N. Vadlamani, "CoPS - Cooperative Provenance System with ZKP Using Ethereum Blockchain Smart Contracts", International Journal of Distributed Systems and Technologies, 2018, pp. 40-53.
[8] https://etherscan.io/
[9] https://ethplorer.io/
[10] https://www.etherchain.org/
[11] F. A. Pratama and K. Mutijarsa, "Query Support for Data Processing and Analysis on Ethereum Blockchain," 2018 International Symposium on Electronics and Smart Devices (ISESD), Bandung, 2018, pp. 1-5.doi: 10.1109/ISESD.2018.8605476 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8605476&isnumber=8605438
[12] M. Muzammal, Q. Qu and B. Nasrulin, "Renovating blockchain with distributed databases: An open source system", Future Generation Computer Systems 90, 2019, pp. 105–117
[13] S. Bragagnolo, H. Rocha, M. Denker and S. Ducasse, "Ethereum Query Language," 2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), Gothenburg, Sweden, 2018, pp. 1-8. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8445051&isnumber=8445042
[14] Web3.JS API. (n.d.). Retrieved from https://web3js.readthedocs.io/en/1.0/
[15] https://docs.eth.events/en/latest/

AUTHORS PROFILE

Navya Gouru Navya is a Research Scholar pursuing Ph.D. on Cloud Security with Blockchain from GITAM (Deemed to be University), India. Navya possess 8 years of IT experience that includes 2 years of experience as a Blockchain Developer writing Smart Contracts and creating DApps. She has experience in working with giant IT companies like Accenture from India and SAP from Singapore and also possess experience in working with startups like ChainNinja as an Ethereum Blockchain Developer and provides PoC's for both public and private Ethereum Blockchain that includes Tokenization and Securing Digital Assets. Navya also has one-year teaching experience for graduates and also a passionate Sign Language Interpreter in ASL, BSL and SEE.

NagaLakshmi Vadlamani has awarded a Ph.D. in computer science and currently working as a Professor in computer science, GITAM Institute of science at GITAM (Deemed to be University), Visakhapatnam, India. Her research interests include Cryptography and Network Security, Database Security and Cloud Computing. She has 20 years of teaching and research experience. She has published more than 20 publications in International and National journals and Proceedings.

doi: 10.32622/ijrat. 77201913