Scientific Journal Impact Factor (SJIF) : 1.711
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745 ISSN (PRINT) : 2393 - 8161

Encryption Techniques for Big Data in a Cloud

Depavath Harinath[1], K.Ramesh Babu[2] ,Mrs.B.Chithra[3], M V Ramana Murthy[4]
[1]*Dept. of Computer Science, HRD Degree and P.G College Narayanaguda, Hyderabad.*
[2]Dept. of Mathematics, M.V.S.R Engineering College, Nadergul,R.R.District,Hyderabad.
[3,4]*Dept. of Computer Science, Osmania University, Hyderabad.*

Abstract— Cloud environment is widely used in industry and research aspects; therefore security is an important aspect for organizations running on these cloud environment. Cloud computing security is developing at a rapid pace which includes computer security, network security, information security, and data privacy. Cloud computing plays a very vital role in protecting data, applications and the related infrastructure with the help of policies, technologies, controls, and big data tools. Moreover, cloud computing, big data and its applications, advantages are likely to represent the most promising new frontiers in science.

Keywords— Cloud Computing, Big Data, Encryption.

I. INTRODUCTION

1.1 Big Data

Big data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed time. Big data "size" is a constantly moving target, as of 2012 ranging from a few dozen terabytes to many petabytes of data. Big data is a set of techniques and technologies that require new forms of integration to uncover large hidden values from large datasets that are diverse, complex, and of a massive scale.

Big data can be described by the following characteristics:

(a.)Volume

The quantity of generated data is important in this context. The size of the data determines the value and potential of the data under consideration, and whether it can actually be considered big data or not. The name 'big data' itself contains a term related to size, and hence the characteristic.

(b.)Variety

This is the category of big data, and an essential fact that data analysts must know. This helps people who analyze the data and are associated with it effectively use the data to their advantage and thus uphold the importance of the big data.

(c.)Velocity

'Velocity' in this context means how fast the data is generated and processed to meet the demands and the challenges that lie in the path of growth and development.

(d.)Variability

This refers to inconsistency the data can show at times—which hampers the process of handling and managing the data effectively.

(e.)Veracity

The quality of captured data can vary greatly. Accurate analysis depends on the veracity of source data.

(f.)Complexity

Data management can be very complex, especially when large volumes of data come from multiple sources. Data must be linked, connected, and correlated so users can grasp the information the data is supposed to convey.

Technologies today not only support the collection of large amounts of data, but also help in utilizing such data effectively. Some of the real time examples of Big data are credit card transactions made

@IJMTER-2015, All rights Reserved

all over the world with respect to a Bank, Walmart customer transactions, and Facebook users generating social interaction data.

Figure: Big Data

When making an attempt to understand the concept of Big Data, the words such as "Map Reduce" and "Hadoop" cannot be avoided.

Hadoop –

Hadoop, which is a free, Java-based programming framework supports the processing of large sets of data in a distributed computing environment. It is a part of theApacheproject sponsored by the Apache Software Foundation. Hadoop cluster uses a Master/Slave structure. Using Hadoop, large data sets can be processed across a cluster of servers and applications can be run on systems with thousands of nodes involving thousands ofterabytes. Distributed file system in Hadoop helps in rapid data transfer rates and allows the system to continue its normal operation even in the case of some node failures. This approach lowers the risk of an entire system failure, even in the case of a significant number of node failures. Hadoop enables a computing solution that is scalable, cost effective, flexible and fault tolerant. Hadoop Framework is used by popular companies like Google, Yahoo, Amazon and IBM etc., to support their applications involving huge amounts of data. Hadoop has two main sub projects – Map Reduce and Hadoop Distributed File System (HDFS).

Map Reduce-

Hadoop Map Reduce is a framework used to write applications that process large amounts of data in parallel on clusters of commodity hardware resources in a reliable, fault-tolerant manner. A Map Reduce job first divides the data into individual chunks which are processed by Map jobs in parallel. The outputs of the maps sorted by the framework are then input to the reduce tasks. Generally the input and the output of the job are both stored in a file-system. Scheduling, Monitoring and re-executing failed tasks are taken care by the framework.

Hadoop Distributed File System (HDFS)-

HDFS is a file system that spans all the nodes in a Hadoop cluster for data storage. It links together file systems on local nodes to make it into one large file system. HDFS improves reliability by replicating data across multiple sources to overcome node failures.

Big data applications-

The big data application refers to the large scale distributed applications which usually work with large data sets. Data exploration and analysis turned into a difficult problem in many sectors in the span of big data. With large and complex data, computation becomes difficult to be handled by the traditional data processing applications which triggers the development of big data applications . Google's map reduce framework and apache Hadoop are the defacto software systems for big data

applications, in which these applications generates a huge amount of intermediate data. Manufacturing and Bioinformatics are the two major areas of big data applications.
Big data provide an infrastructure for transparency in manufacturing industry, which has the ability to unravel uncertainties such as inconsistent component performance and availability. In these big data applications, a conceptual framework of predictive manufacturing begins with data acquisition where there is a possibility to acquire different types of sensory data such as pressure, vibration, acoustics, voltage, current, and controller data. The combination of sensory data and historical data constructs the big data in manufacturing. This generated big data from the above combination acts as the input into predictive tools and preventive strategies such as prognostics and health management.
Another important application for Hadoop is Bioinformatics which covers the next generation sequencing and other biological domains. Bioinformatics which requires a large scale data analysis, uses Hadoop. Cloud computing gets the parallel distributed computing framework together with computer clusters and web interfaces.

Big data advantages-

In Big data, the software packages provide a rich set of tools and options where an individual could map the entire data landscape across the company, thus allowing the individual to analyze the threats he/she faces internally. This is considered as one of the main advantages as big data keeps the data safe. With this an individual can be able to detect the potentially sensitive information that is not protected in an appropriate manner and makes sure it is stored according to the regulatory requirements.
There are some common characteristics of big data, such as
a) Big data integrates both structured and unstructured data.
b) Addresses speed and scalability, mobility and security, flexibility and stability.
c) In big data the realization time to information is critical to extract value from various datasources, including mobile devices, radio frequency identification, the web and a growing list of automated sensory technologies.

All the organizations and business would benefit from speed, capacity, and scalability of cloud storage. Moreover, end users can visualize the data and companies can find new business opportunities. Another notable advantage with big-data is, data analytics, which allow the individual to personalize the content or look and feel of the website in real time so that it suits the each customer entering the website. If big data are combined with predictive analytics, it produces a challenge for many industries. The combination results in the exploration of
these four areas:
a) Calculate the risks on large portfolios
b) Detect, prevent, and re-audit financial fraud
c) Improve delinquent collections
d) Execute high value marketing campaigns

Need of security in big data-

For marketing and research, many of the businesses uses big data, but may not have the fundamental assets particularly from a security perspective. If a security breach occurs to big data, it would result in even more serious legal repercussions and reputational damage than at present. In this new era, many companies are using the technology to store and analyze petabytes of data about their company, business and their customers. As a result, information classification becomes even more critical. For making big data secure, techniques such as encryption, logging, honeypot detection must be necessary. In many organizations, the deployment of big data for fraud detection is very attractive and useful.
The challenge of detecting and preventing advanced threats and malicious intruders, must be solved using big data style analysis. These techniques help in detecting the threats in the early stages using more sophisticated pattern analysis and analyzing multiple data sources.

@IJMTER-2015, All rights Reserved

Not only security but also data privacy challenges existing industries and federal organizations. With the increase in the use of big data in business, many companies are wrestling with privacy issues. Data privacy is a liability, thus companies must be on privacy defensive. But unlike security, privacy should be considered as an asset, therefore it becomes a selling point for both customers and other stakeholders. There should be a balance between data privacy and national security.

1.1 Cloud Computing

Cloud Computing is a technology which depends on sharing of computing resources than having local servers or personal devices to handle the applications. In Cloud Computing, the word "Cloud" means "The Internet", so Cloud Computing means a type of computing in which services are delivered through the Internet. The goal of Cloud Computing is to make use of increasing computing power to execute millions of instructions per second. Cloud Computing uses networks of a large group of servers with specialized connections to distribute data processing among the servers. Instead of installing a software suite for each computer, this technology requires to install a single software in each computer that allows users to log into a Web-based service and which also hosts all the programs required by the user. There's a significant workload shift, in a cloud computing system.

Local computers no longer have to take the entire burden when it comes to running applications. Cloud computing technology is being used to minimize the usage cost of computing resources. The cloud network, consisting of a network of computers, handles the load instead. The cost of software and hardware on the user end decreases.

The only thing that must be done at the user's end is to run the cloud interface software to connect to the cloud. Cloud Computing consists of a front end and back end. The front end includes the user's computer and software required to access the cloud network. Back end consists of various computers, servers and database systems that create the cloud. The user can access applications in the cloud network from anywhere by connecting to the cloud using the Internet. Some of the real time applications which use Cloud Computing are Gmail, Google Calendar, Google Docs and Dropbox etc.,

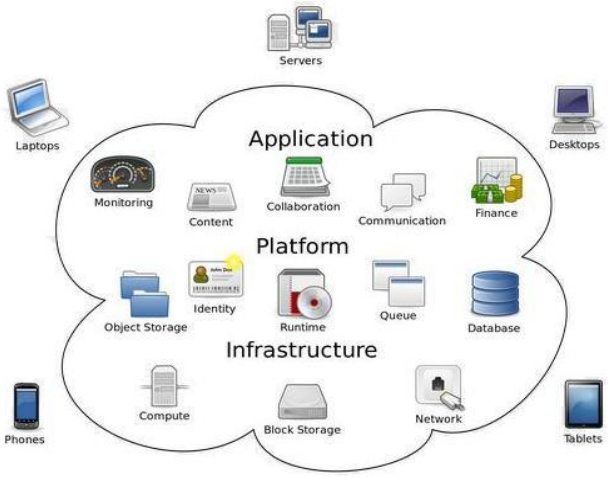

Figure1: Cloud computing

As the cloud has evolved, there are many compelling reasons to migrate applications and data to private or public clouds: scalability, agility, cost savings. But along with the benefits, the risk to the safety of user data and business-critical data has increased due to the lack of security techniques to protect the large volume of the data. Security and privacy of data stored in the cloud are major challenges and key issues for the cloud.

Encryption is a well-known technique for preserving the privacy of sensitive information. It is being adopted in many secure cloud computing applications. Encryption technology, however, limits functionality and is the basic and inherent drawback. While data can be sent to and from a cloud provider's data center in encrypted form, the servers that power a cloud cannot do any work on it. An information system working with encrypted data can, at most, store or retrieve the data for the user. Any more complicated operations require that the data be decrypted beforehand.

The increased demand for enhanced security and inherent limits of traditional encryption schemes for data on the cloud has driven the adoption and implementation of several new encryption algorithms such as searchable, order-preserving, and homomorphic schemes. Searchable encryption allows us to encrypt data in such a way that it can still be searched. It enables a user to securely outsource data to an untrusted cloud provider without sacrificing the ability to search over it. An order preserving encryption scheme is an encryption algorithm that produces ciphertexts while preserving numerical ordering of the plaintexts. It is an important technique for database related applications due to its capability of supporting range query processing directly on encrypted data without needing to decrypt them. Homomorphic encryption allows us to perform a mathematical operation on the encrypted data, and produces the same answer as performing an analogous operation on the unencrypted data.
This paper investigates encryption schemes currently implemented in many secure products in cloud environments. Each encryption scheme is evaluated and analyzed in its efficiency, security and functionality for big data on the cloud.

II. LITERATURE SURVEY

A . Advanced Encryption Standard

NIST established the Advanced Encryption Standard (AES) in 2001 [18] as an approved standard for a wide range of applications. AES is a symmetric key algorithm in which the same key is used for both encryption and decryption of data. AES is extensively used in practical secure applications for data in the cloud as shown in table 1.

While AES is a widely-adopted encryption scheme for data on cloud storage, it limits many application functionalities such as search, logical operations and mathematical calculation. Scalability of AES in cloud environments is also a significant issue to be addressed.

Table I . Current cryptosystems

Current Cryptosystems	
Product	***Encryption***
Cloudera, Navigator Encrypt	AES-256
SafeNet, ProtectDB	AES, 3DES, DES, RSA, RC4, SHA-1, ACSHA-1
Thales, Hardware Security Modules (HSM)	AES (128, 192,256) 3DES, RSA ECC
CloudLink RSA Data Protect Manager	AES - 256
HP, Altlla	AES

B. Public Key Cryptography

Introduced in the pioneering paper by Diffie and Hellman [36], public key cryptography provides the foundation for both key management and digital signatures. In key management, public key cryptography is used to distribute the secret keys used in other cryptographic algorithms (e.g. AES). For digital signatures, public key cryptography is used to authenticate the origin and protect the integrity of data. It is, however, well known that public-key cryptography demands considerable computing resources. For data in cloud storage, this problem is
exacerbated and one may ask if it is possible at all to implement public-key cryptography efficiently on cloud storage.

1) Rivest-Shamir-Adleman scheme

The Rivest-Shamir-Adleman (RSA) scheme [26] is one of the first successful cryptographic algorithm that met the requirements for public-key systems and the most widely accepted and implemented general-purpose approach to public-key encryption. The RSA scheme takes the plaintext and ciphertext with integers between 0 and *n* - 1 for some *n*. The US National Institute for Standards and Technology has recommended that these systmes, typically using 1024 bits, are sufficient for use until 2010.

Since RSA is based on arithmetic modulo of large numbers it can be slow in constrained environments. There are some variants of RSA proposed to improve the performance, such as Batch RSA, Multi-factor RSA, or Rebalanced RSA. Fiat [3] observed that, when using small public exponents, it is possible to decrypt two ciphertexts for approximately the price of one. This batching technique is only worthwhile when the public exponents are small (e.g., 3 and 5). Otherwise, the extra arithmetic required is too expensive. Also, one can only batchdecrypt ciphertexts encrypted using the same modulus and distinct public exponents. Fiat generalized the above observation to the decryption of a batch of RSA ciphertexts. The multi-factor RSA is based on modifying the structure of the RSA modulus. The two multi-factor RSA techniques are promising in that they are fully backwards compatible.[31, 33]

In standard RSA, encryption and signature verification are much less processor-intensive than decryption and signature generation. The rebalanced RSA enables us to rebalance the difficulty of encryption and decryption, based on Wiener [21].Table II gives the speedup factors for each of these variants using a 1024-bit RSA modulus.

Table II: Comparison of RSA variants

Method	*Speedup*
Batch RSA	2.64
Multi-prime	2.25
Multi-power	3.38
Rebalanced	3.20

2) Elliptic Curve Cryptography:

Elliptic Curve Cryptography (ECC) is based on the mathematical theory of elliptic curves [28, 4]. It can provide the same level and type of security as RSA but with much shorter keys. NSA compares the key sizes for three different approaches to encryption for comparable levels of security against brute-force attacks [37]. Key size comparison indicates that, with ECC, it takes one-sixth the computational effort to provide the same level of cryptographic security that you get with 1024-bit RSA. Because of the much smaller key sizes involved, ECC algorithms can be implemented on smartcards without mathematical coprocessors. Contactless smart cards work only with ECC because other systems require too much induction energy. Since shorter key lengths translate into faster handshaking protocols, ECC is also becoming increasingly important for wireless communications.

@IJMTER-2015, All rights Reserved

C. Searchable Encryption

1) Searchable Symmetric Encryption

Symmetric searchable encryption (SSE) is appropriate in any setting where the party that searches over the data is also the one who generates it. SSE schemes were introduced in [14] and improved constructions and security definitions are given in [15]. The main advantages of SSE are efficiency and security, while the main disadvantage is functionality. Song, Wagner, and Perrig proposed the first practical scheme for searching encrypted data [14], achieved using a special two-layered encryption construct for each word that allows searching the ciphertexts with a sequential scan. Goh [15] addresses some of the limitations (e.g., use of fixed-size words, special document encryption) by adding an index for each document, which is independent of the underlying encryption algorithm. He defines a secure index and formulates a security model for indexes known as semantic security against adaptive chosen keyword attack (IND-CKA). He also shows how to construct an efficient IND-CKA secure index called Z-IDX using pseudo- random functions and Broom filter.By extending the techniques in [15], Chang and Mitzenmacher [38] develop two secure index schemes (CM-I,
CM-II) using pre-built dictionaries. The idea is to use a prebuilt dictionary of search keywords to build one index per document. The index is an m-bit array, initially set to 0, where each bit position corresponds to a keyword in the dictionary. If
the document contains a keyword, its index bit is set to 1. CMI and CM-II assume that the user is mobile with limited storage space and bandwidth, so the schemes require only a small amount of communication overhead. Both constructions use only pseudo-random permutations and pseudo-random functions. CM-I stores the dictionary at the client and CM-II encrypted at the server. Both constructions can handle secure updates to the document collection in the sense that CM-I and CM-II ensure the security of the consequent submissions in the presence of previous queries.

2) Public Key Encryption with Keyword Search

Public Key Encryption with keyword search schemes areappropriate in any setting where the party searching over the data is different from the party that generates it. Public Key Encryption with keyword search schemes were introduced in[13] while improved definitions were given in [11]. Several works have shown how to achieve more complex search queries in the public-key setting, including conjunctive searches and range queries. The main advantage of searchable
public-key encryption is functionality, while the main disadvantages are inefficiency and weaker security guarantees.
Since the writer and reader can be different, searchable publickey encryption schemes can be used in a larger number of settings than SSE schemes.

3) Deterministic Encryption

Deterministic encryption schemes are cryptosystems which always produces the same ciphetext for a given plaintext and key, even over separate executions of the encryption algorithm. Deterministic public-key encryption was introduced as an alternative in scenarios where randomized encryption has inherent drawbacks. Ciphertexts produced by randomized encryption algorithm are not length preserving, and generally not efficiently searchable. These properties are problematic in many applications involving massive amount of data.
Recent research [20] shows that a natural variant of the notion of semantic security can be guaranteed even when using deterministic encryption algorithm, as long as plaintexts are somewhat unpredictable, and independent of the public key used by the scheme Deterministic encryption has two inherent limitations. First, no privacy is possible if the plaintext is known to come from a small space. This can be addressed by requiring that the plaintext is drawn from a space of large min-entropy. Second, the ciphertext itself is partial information about the plain text. When the plaintext and partial information do not depend on the public key, there would be no leakage of partial information.

@IJMTER-2015, All rights Reserved

D. Order Preserving Encryption Scheme

Traditional encryption techniques do not preserve numerical order of the plaintexts. The integration of existing encryption techniques with database systems has problems with performance degradation, since database indices such as B-tree can no longer be used. An order preserving encryption scheme (OPES) is a deterministic encryption scheme whose encryption algorithm produces ciphertexts that preserve numerical ordering of the plaintexts. The basic idea of the scheme is to take as input a user-provided target distribution and transform the plaintext values in such a way that the transform preserves the order, while the transformed values follow the target distribution [24]. Order preserving encryption scheme are important techniques for database related applications due to their capacity for supporting range query processing directly on encrypted data without needing to decrypt them. It can also easily be integrated with existing database systems as it has been designed to work with the existing indexing structures, such as B-trees. OPES allows comparison operations to be directly applied on encrypted data, without decrypting the operand.

G. Ozsoyoglu, D. Singer, and S. Chung [16] considered attribute (field)-level encryption for relational databases and showed that, given a sequence of nonnegative integers, there is a uniquely determined order preserving function. A sequence of strictly increasing polynomial functions is used for the input distribution function. The shape of the distribution of encrypted values depends on the shape of input distribution because this encryption method does not take the input distribution into account. This scheme may reveal information about the input distribution, which can be exploited. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu [24] has designed OPES that remove the dependency of the result of encryption on the input distribution. The result of encryption in this scheme is statistically indistinguishable. They introduced "Flatten" stage to make the result of encryption is statistically indistinguishable. R. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan [23] present CryptDB, a system that explores an intermediate design point to provide confidentiality for applications that use database management systems (DBMSes). CryptDB leverages the typical structure of database-backed applications, consisting of a DBMS server and a separate application server; the latter runs the application code and issues DBMS queries on behalf of one or more users. CryptDB's approach is to execute queries over encrypted data, and the key insight that makes it practical is that SQL uses a well-defined set of operators, each of which we are able to efficiently support over encrypted data. An analysis of a trace of 126 million SQL queries from a production MySQL server shows that CryptDB can support operations over encrypted data for 99.5% of the 128,840 columns seen in the trace [17]. Formal cryptographic treatment of OPES did not appear until recently, in the paper by A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill [2, 1].

E. Homomorphic Encryption

Homomorphic encryption allows us to encrypt the data in such a way that performing a mathematical operation on the encrypted data and decrypting the result produces the same answer as performing an analogous operation on the unencrypted data [26]. The correspondence between the operations on unencrypted data and the operations performed on encrypted data is known as a homomorphism. In principle, something like this could be used to secure operations over the cloud. A cryptosystem is considered partially homomorphic if it exhibits either additive or multiplicative homomorphism, but not both. There are many forms of partially homomorphic cryptosystems that allow for some specific operations to be performed [26, 31, 21]. Recently, Craig Gentry proposed fully homomorphic cryptosystem [7].

1) Fully Homomorphic Encryption

In 2009, Craig Gentry developed the first fully homomorphic cryptosystem. Rather than using simple modular arithmetic like most other cryptosystems, Gentry's cryptosystem is lattice-based. The Gentry scheme relies on a complex mesh of ideal lattices for representing the keys and the ciphertext. The fact that it is lattice-based makes implementation very complex and operations run very slowly on the ciphertext. Additionally, tuning the cryptosystem so that it exhibits better

performance drastically reduces security of the system and even prevents homomorphism in some circumstances [7]. Gentry acknowledges that the system runs too slowly for practical use and estimates that these applications could be ready for the market in five to 10 years. Gentry and Halevi [8] described the first implementation of Gentry's scheme, using many clever optimizations including some suggested in a previous work by Smart and Vercauteren [48]. For their most secure setting (claiming 72- bit security) the authors report a public key size of 2 ~3 GB and a ciphertext refresh procedure taking 30 minutes on a high-end workstation. Brakerski and Vaikuntanathan's scheme is based on the Learning with Errors (LWE) and Ring Learning with Errors (RLWE) problems [38, 39]. These authors introduce a new dimension reduction technique and a new modulus switching technique to shorten the ciphertext and reduce the decryption complexity. Brakerski, Gentry, and Vaikuntanathan [41] introduced a remarkable new FHE framework, in which the noise ceiling increases only linearly with the multiplicative level instead of exponentially. This implies that bootstrapping is no longer necessary to achieve fully homomorphic encryption. This new framework has the potential to significantly improve the practical FHE performance. The new framework is based on Brakerski and Vaikuntanathan's scheme [38, 39], and more specifically on their new modulus switching technique, which efficiently transforms a ciphertext encrypted under a certain modulus p into a ciphertext under a different modulus p' but with reduced noise.

2) Fully Homomorphic Encryption over the Integers

The DGHV (Van **D**ijk, **G**entry, **H**alevi and **V**aikuntanathan) scheme over the integers [33] is conceptually simpler than Gentry's scheme, because it operates on integers instead of ideal lattices. They considered Fully Homomorphic Encryption over Integers. For the encryption of arbitrary messages and the homomorphic evaluation of arbitrary functions on ciphertexts, it is in fact sufficient to construct scheme to encrypt single bit messages (either 0 or 1) and evaluate an arbitrary number of XOR and AND logic gates on encryptions of those bits. Indeed, data of any length can be represented as a bit string, and arbitrary functions on such a bit string can be represented as Boolean circuits (consisting of XOR and AND gates) on the corresponding bits. There are three major drawbacks that hold back the performance of fully homomorphic encryption over the integer: ciphertext expansion, overhead of homomorphic evaluation, and the size of the public key and of public parameters. Ciphertexts consists of millions of digits being used for a single message bit. Evaluating an operation as simple as a bitwise AND requires carrying out exact arithmetic on those huge integers, which is slow and expensive. The conversion from secret to public key for a fully homomorphic scheme can be done in a relatively straightforward manner. It is sufficient to publish a large number of encryptions of 0, but this results in a prohibitively large public key. Moreover, the bootstrapping method requires publishing very large public parameters for homomorphic evaluation, even for secret key schemes. Recent work in [42] improved the performance by reducing the public keys and parameters down to only a few megabytes. It increased the encryption speed and enables us to obtain a proof of concept implementation executable in reasonable time on an ordinary computer. Authors in [43] proposed yet another fully homomorphic encryption scheme over the integers offering dramatic improvements to both ciphertext expansion and homomorphic evaluation overhead at the same time. Improvements in [19] avoid the use of the expensive bootstrapping method when evaluating arbitrary functions homomorphically. With that change, homomorphic AND operations only increase size of ciphertext noise by a small fixed amount instead of doubling it every time. These results improve the speed of fully homomorphic encryption of integers by about two order of magnitude.

3) Homomorphic evaluation of AES

Homomorphic evaluation of AES decryption has an interesting application: "When data is encrypted under AES and we want to compute on that data, homomorphic AES decryption would transform this AES encrypted data into fully homomorphic encrypted data those we could perform whatever computation we want."[44]. Fig.1 illustrates the AES based Homomorphic encryption. Data can be

sent encrypted under AES along with the public key $\mathbf{K}_H$ of the FHE scheme as well as the AES secret key encrypted under **K**H. Then, before the cloud performs homomorphic operations on the data, it can first run the AES decryption algorithm homomorphically to obtain the plaintext data encrypted under $\mathbf{PK}_{FHE}$.

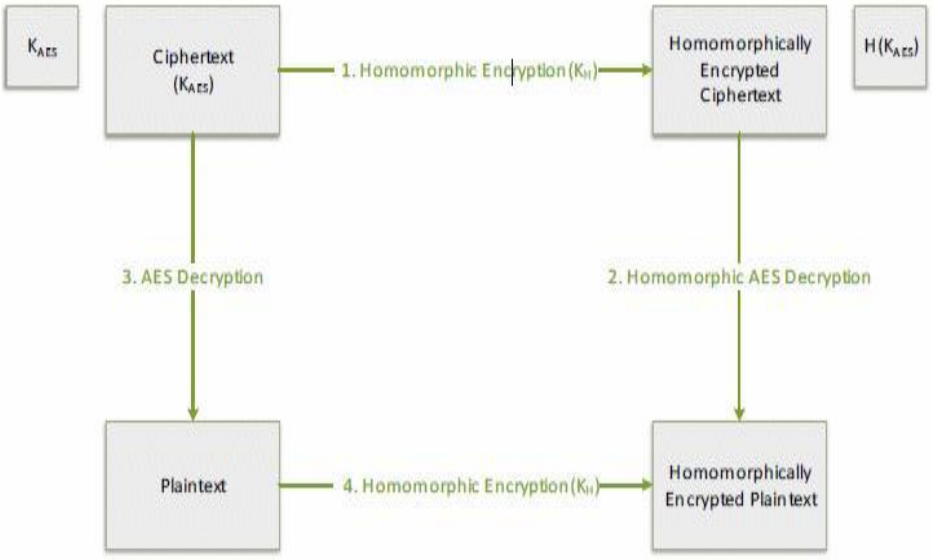

Figure 3. Homomorphic encryption using AES

Homomorphic evaluation of the AES circuit had been studied with leveled homomorphic encryption without bootstrapping [9]. The implementation is based on the NTL C++ library running over GMP, and a machine that consists of a processing unit of Intel Xeon CPUs running at 2.0GHz with 18MB cache with 256GB of RAM. Using SIMD techniques, they processed over 54 blocks in each evaluation, yielding an amortized rate of less than 40 minutes per block. Another implementation process 720 blocks in each evaluation, yielding an amortized rate of over five minutes per block. A recent study [19] used a variant of Simple Fully Homomorphic Encryption scheme over the integers with the scale-invariant property as in [39]. They also adapt the batch setting and homomorphically evaluate AES encryption. Their scheme offers competitive performances as it can evaluate the full AES circuit in about 23 seconds per AES block at the 72-bit security level.

F. Key Management

In order for encryption to work effectively, it is important to manage the encryption keys securely. Even if a cloud service provider provides encryption, they might access the keys. When encrypted data is stored in the cloud, the keys used for encryption should be kept separate and should only be accessed by the end user. Key management involves the creation, use, distribution, and destruction of encryption keys.

1) Traditional Key Management

The management of the keys is what prevents them from falling into the wrong hands. There are three categories of traditional encryption key management solutions.

a) Key Management Server in the Data Center

b) Key Management by the Cloud Provider

c) Key Management by a "Third Party" Provider

The first option gives good control over the confidentiality of keys, but it adds operational overhead, is expensive, and is far from flexible, scalable, or elastic. In the second and third options we lose ownership of data, leaving us with no control and no regulatory compliance because someone else now has access to encryption keys. To overcome existing key management systems for data on cloud, new key management systems based on recent encryption schemes, identity-based encryption [12] and homomorphic encryption, were proposed.

@IJMTER-2015, All rights Reserved

The following sections review two of them.

2) Stateless Key Management

Voltage Security® provides Stateless Key Management that enables on-demand key generation and re-generation without an ever-growing key store. The result is a system that can be infinitely scaled across distributed physical and logical locations with no additional overhead [35]. Stateless Key Management is based on the identity-based encryption that takes a breakthrough approach to the problem of encryption key management. Identity-based encryption can use any arbitrary string as a public key, enabling data to be protected without the need for certificates. Protection is provided by a key server that controls the dynamic generation of private decryption keys corresponding to public identities. The stateless nature of identity-based encryption also dramatically simplifies operation and scaling. Key Servers can be distributed independently and geographically, and key requests can be load balanced across them without the need to synchronize data. This enables a larger scale without growing complexity and allows for distributed and federated key management across the world easily and quickly.

3) Homomorphic Key Management System

Porticor has developed a patent-pending technology that implements homomorphic techniques for combining and splitting encryption keys [46]. It enables the Porticor Virtual Appliance to give the application access to the data store without ever exposing the master keys in an unencrypted state. The security of the HKM protocol relies on the semantic security of ElGamal encryption with a key size of 2048 bits. Functionality of the protocol builds on the multiplicative homomorphic properties of ElGamal encryption. Authenticity and secrecy of communication is obtained using SSL, which means that a trusted PKI has to be in place.

III. ENCRYPTION TECHNIQUE SCHEME

A. AES encryption scheme

Implementation of ASE scheme depends on key size, block size and mode of operations.

1*) Key size and block Size*

The key length for AES can be 128, 192, or 256 bits, and are referred to as AES-128, AES-192, or AES-256, respectively. Generally, a longer key size results in a more secure scheme; however, longer key sizes need more computing resources and may degrade the performance of the scheme. AES uses block sizes of 128 bits though it supports other block sizes also.

2) Supporting mode of operations

There are six cryptographic modes of operation for symmetric encryption. Even though modes are standardized and well-known, their effectiveness varies [45]. Different block cipher modes of operation can have significantly different performance and efficiency characteristics, even when used with the same block cipher. Counter (CTR) mode is an initial vector-based encryption scheme. The mode achieves indistinguishability from random bits assuming a nonce initialization vector (IV). Because of the parallelizability of the mode, it is faster - in some settings much faster - than other modes. Cipher Block Chaining (CBC) is a popular block cipher mode operation where each plaintext is XORed with the previous ciphertext block before being encrypted. Hence, each ciphertext is dependent on all plaintext blocks up to that stage. However, ciphertexts are highly malleable and vulnerable to a chosen-ciphertext attack (CCA). Confidentiality is forfeit in the presence of a correct-padding oracle for many padding methods, and encryption is inefficient from being inherently serial. Though widely used, the mode's privacy-only security properties result in frequent misuse. Galois/Counter Mode (GCM) is a mode of operation that has been widely adopted because of its efficiency and performance. GCM can take full advantage of parallel processing, and an implementation can make efficient use of instruction or hardware pipelines.

@IJMTER-2015, All rights Reserved

B. Fully Homomorphic Encryption over the Integers

We use the following simple symmetric encryption scheme in [34]:

KeyGen: The key is an odd integer, chosen from some Interval $p \in (2^{\eta-1}, 2^{\eta})$, where η is the bit length of the key. **Encrypt*(p,m)*** : To encrypt a bit $m \in \{0,1\}$, set the ciphertext as an integer whose residue has the same parity as the plaintext. Namely, set c=pq+2r+m, where the integers q,r are chosen at random in some other prescribed intervals, such that 2r is smaller than p/2 in absolute value.

Decrypt*(p,c)* : Output *(c mod p) mod 2*

All input parameters are polynomials of security parameter λ. For a fixed value of the security parameter, the size of the secret key is always the same; similarly, all the ciphertexts that can be decrypted have the same size.

IV. EVOLUTION AND DISCUSSION

EVALUATION -

A. AES encryption scheme

We evaluated AES encryption scheme in different modes of operations, CBC and CTR, with key sizes 192 and 256 bits. We measure the speedas well as CPU usage for different sizes of data and compared the results. This analysis will be a groundwork for the same analysis and experiments for the homomorphic evaluation of AES scheme and allow us to compare results to those obtained using the "traditional" AES. Figure below showed the tradeoff between block sizes as well as key sizes. We can visualize that the encryption as well as decryption uses less CPU times when we reduce the key size. When we use a smaller key size, the XOR operation with plaintext with the key stream is much faster. The difference is obvious when the size of the file was big (~ 500 MB). For small sizes of file, the dependence on the key size is negligible.

The similar case goes if the block size is large enough, the process of encryption is faster compared to encryption with a smaller block size. Supposing we use a block size of 16 bytes, the time it takes to chunk the file into a block of 16 bytes and then XOR with a key size large than the block size gets extra bits of data. They would need to be discarded in normal operation, although in CTR they are avoided. The kinds of operations make the computation a bit harder. So we need a fixed correct block size and key size to optimize the performance.

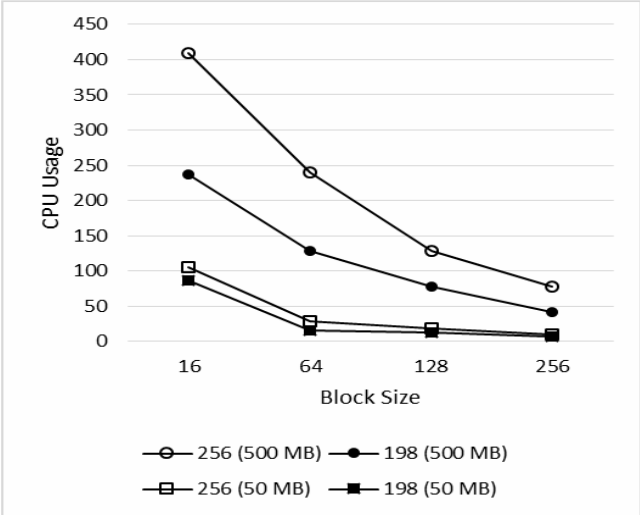

Figure :. CPU usage with different file sizes and block sizes in AES

Encryption code is rarely thread safe, as it requires complex mathematics to generate secure output. AES is relatively fast; if you need better speed from it, consider vertical scaling or hardware accelerators as a first step. Later, you can add more servers to encrypt different files concurrently (horizontal scaling). But since AES CTR is an individual operation whose cipher texts does not

affect another block of encryption, we can achieve more performance in CTR. In normal AES, thread creation does not make any difference because JAVA JVM automatically creates thread. But in modes like CBC, the operation involving the encrypted block needs to be XORed with next block of plaintext. So thread operation doesn't make any difference.

B. Parallel implementation of AES encryption scheme

Figure below shows the calculated execution time of AES encryption according to the "parallel implementation" proposed in [42]. The content document for encryption (nsquare of information) is perused at once where n ought to be more noteworthy number, for example 1000, 2000, 3000 and so forth, to accomplish the better execution. First n/2 blocks can be relegated to the one core (C1) for encryption/decoding, while an alternate n/2 blocks can be allocated to other core (C2) for performing encryption/decryption. For this situation we are performing encryption/decryption on various blocks of information by using the idea of synchronous multithreading a percentage of the blocks by core C1 and a portion of the pieces by core C2. This methodology will proceed until the end of the record and, after the last encryption/decryption step, give the execution (encryption and decryption) time for the whole document.

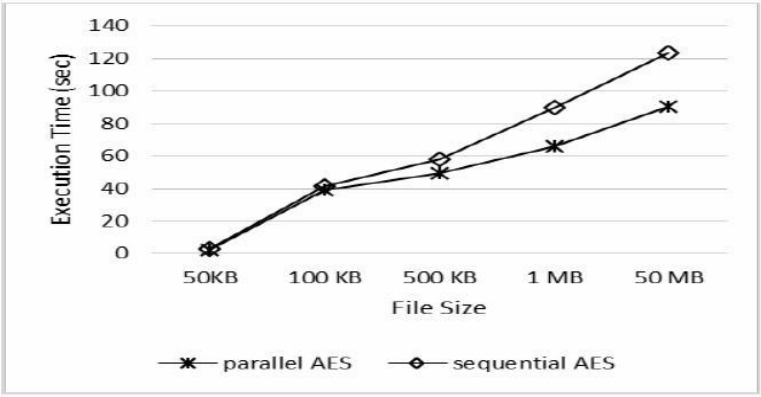

Figure . Execution time of AES using multithreading.

Execution time is compared to that obtained using AES-CBC implementation. As the size of the file increases, parallel implementation improves execution time.

C. Fully Homomorphic Encryption over the Integers

Table III shows our results of encryption and decryption time for the simple fully homomorphic encryption [33] with differentsecurity parameter size based on a C++ implementation using CryptoCpp/GMP library, on a machine which consists of a processing unit of Intel Code i5-3320M CPUs running at 2.60GHz with 4GB of RAM.

TABLE III. Results of Encryption and of Encryption and decryption time for the simple fully homomorphic scheme with different size of Security parameter.

Security Parameter (λ)	*Key Generation (sec)*	*Encrypt/ Decrypt 100 bit (sec)*	*Encrypt/ Decrypt 200 bit (sec)*
2	0.000232	0.000666	0.001139
3	0.002444	0.05991	0.105942
4	0.109338	6.180701	12.099349
5	5.881542	422.310419	801.081168

DISCUSSIONS-

In Fig. 4, we consider the various encryption techniques and compiled a graph that approximately ranks the security of different encryption types, functionality, and efficiency of each. We will discuss these tradeoffs with different encryption techniques.

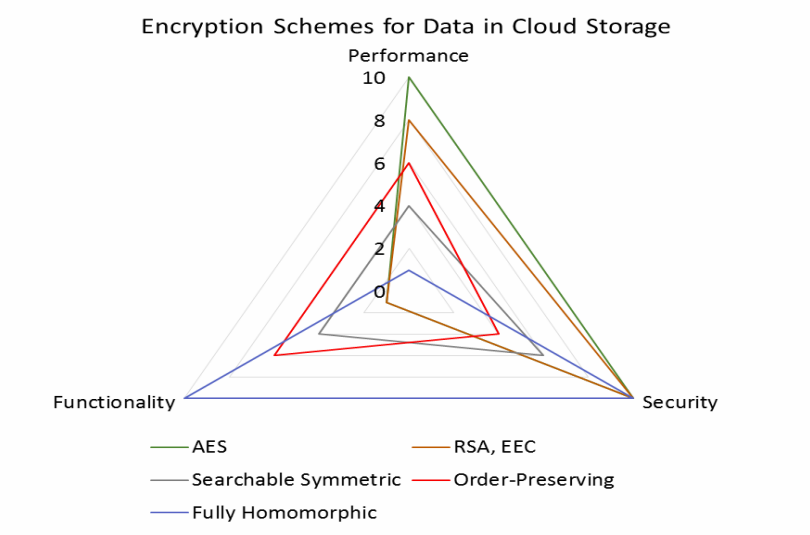

Figure : Tradeoff of encryption schemes for cloud storage

Symmetric encryption schemes based on AES are widely used in existing secure applications for data in the cloud. The popularity of the scheme is due to its efficiency and proven security. However, there are concerns in security and efficiency in cloud environments. One concern is the secure and scalable key management system. Another is how to implement many functionalities promised by cloud computing environments. As an example, an end user encrypts data and saves them in a cloud storage. When the user is going to execute logical operations or statistical analysis, she has to retrieve and decrypt her data, execute operations, re-encrypt, and then send it back to the storage server. Public key encryption schemes demand considerable computing resources and their usage is limited for distributing secret keys used either in other cryptographic algorithms (e.g. AES) or for digital signatures. When the amount of transmitting data is limited and many users are involved, such as in mobile environments, public key encryption schemes (RSA or ECC) have advantages compared to AES in terms of scalability and key management. Fully homomorphic encryption schemes offer various advantages promised by cloud computing. Even though many studies have been devoted towards improvement, performance is still too slow to be practical in secure products. Implementing homomorphic encryption with AES has interesting applications and deserves more research and investigations. “Search” is a key operation demanded for data on cloud storage. Many searchable

@IJMTER-2015, All rights Reserved

encryption schemes have been proposed: symmetric searchable encryption, public key encryption with keyword search, and deterministic encryption scheme. Order preserving encryption schemes are one of searchable encryption for data on relational databases and executed search operation on encrypted data in comparable performance, though preserving order sacrifices security of the encryption scheme. Formal cryptographic treatment of OPES has not appeared until recently.

V. CONCLUSION

Cloud computing plays a very vital role in protecting data, applications and the related infrastructure with the help of policies, technologies, controls, and big data tools. This paper depicts a number of symmetric, public key and homomorphic cryptosystems to help practitioners understand encryption schemes for data on cloud storage. AES is used in most secure applications for data on cloud storage. Fully homomorphic encryption schemes are promising for cloud environment but are far from being practical because of their performance. Homomorphic evaluation of AES has interesting applications as a practical encryption scheme for data on cloud storage. It will be a future work we need to investigate and implement.

REFERENCES

[1] A. Boldyreva, N. Chenette, and A.O'Neill.*Orderpreserving encryption revisited: Improved security analysis and alternative solutions.* In P. Rogaway, editor,CRYPTO, volume 6841 of Lecture Notes in Computer Science, pages 578 ~ 595, 2011.

[2] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill.*Order-preserving symmetric encryption.* In A. Joux, editor, EUROCRYPT, volume 5479 of Lecture Notes in Computer Science, pages 224 ~ 241. Springer, 2009

[3] A. Fiat. *Batch RSA.* In G. Brassard, ed., Proceedings of Crypto 1989, vol. 435 of LNCS, pp. 175–185. Springer-Verlag, Aug. 1989.

[4] A. Jurisic, A. Menezes. *Elliptic Curves and Cryptography*. r. Dobb's Journal, April 1997.

[5] A. Shamir, *Identity-Based Cryptosystems and Signature Schemes*, Proceedings of CRYPTO '84, LNCS 196, pages 47-53, Springer-Verlag, 1984.

[6] C. Cocks, *An Identity Based Encryption Scheme Based on Quadratic Residues*, Cryptography and Coding – Institute of Mathematics and its Application International Conference On Cryptography and Coding – Proceedings of IMA 001, LNCS 2260, pages 360-363, Springer-Verlag 2001.

[7] C. Gentry, *Fully homomorphic encryption using ideal lattices*, Symposium on the Theory of Computing (STOC), 2009, pp. 169-178.

[8] Craig Gentry and Shai Halevi, *Implementing Gentry's fully-homomorphic encryption scheme*, Advances in Cryptology–EUROCRYPT 2011, pp. 129–148, 2011.

[9] Craig Gentry, Shai Halevi, and Nigel P. Smart. *Homomorphic evaluation of the AES circuit*. In Reihaneh.

[10] Craig Gentry. *Fully homomorphic encryption using ideal lattices.* In Michael Mitzenmacher, editor, STOC, pages 169{178. ACM, 2009.

[11] D. Boneh and B. Waters. *Conjunctive, subset, and range queries on encrypted data*. In Theory of Cryptography Conference (TCC '07), volume 4392 of Lecture Notes in Computer Science, pages 535–554. Springer, 2007.

[12] D. Boneh and M. Franklin, *Identity-Based Encryption from the Weil Pairing*, Proceedings of CRYPTO 2001, LNCS 2139, pages 213-229, Springer-Verlag, 2001.

[13] D. Boneh, G. di Crescenzo, R. Ostrovsky, and G.Persiano. *Public key encryption with keyword search*. In C. Cachin and J. Camenisch, editors, Advances in Cryptology – EUROCRYPT '04, volume 3027 of Lecture Notes in Computer Science, pages 506–522. Springer,2004.

[14] D. X. Song, D. Wagner, and A. Perrig. *Practical techniques for searches on encrypted data*. In 2600 IEEE Symposium on Security and Privacy, pages 44–55, Oakland, California, USA, May 2000. IEEE Computer Society Press

[15] E.-J. Goh. *Secure indexes*. Cryptology ePrint Archive, Report 2003/216, 2003.

[16] G. Ozsoyoglu, David Singer, S. Chung. *Anti-tamper databases: Querying encrypted databases*. In Proc. of the 17th Annual IFIP WG 11.3 Working Conference on Database and Applications Security, Estes Park, Colorado, August 2003.

[17] http://css.csail.mit.edu/cryptdb/

[18] J. Daemen, V. Rijmen. Rijndael: *The Advanced Encryption Standard*. Dr. Dobb's Journal, March 2001.

[19] J.-S. Coron, T. Lepoint, and M. Tibouchi: *Scale-invariant Fully Homomorphic Encryption over the Integers*, PKC 2014, LNCS 8383, pp. 311–328, 2014.

[20] M. Bellare, A. Boldyreva, and A. O'Neill. *Deterministic and efficiently searchable encryption.* In A. Menezes, editor, Advances in Cryptology – CRYPTO '07, Lecture Notes in Computer Science, pages 535–552. Springer, 2007.

@IJMTER-2015, All rights Reserved

[21] M. Wiener. Cryptanalysis of Short RSA Secret Exponents. IEEE Trans. Information Theory 36(3):553– 558. May 1990
[22] P. Paillier *Public-key cryptosystems based on composite degree residuosity classes*. In Advances in cryptology—EUROCRYPT'99, pp. 223-238. Springer Berlin Heidelberg, 1999. pages 850-867. Springer, 2012.
[23] R. A. Popa, N. Zeldovich, and H. Balakrishnan.*ryptDB: A practical encrypted relational DBMS.* Technical MITCSAIL-TR-2011-005, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, January 2011.
[24] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. *Orderpreserving encryption for numeric data*. In SIGMOD '04, pp. 63 ~ 574. ACM, 2004.
[25] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. *Searchable symmetric encryption: improved definitions and efficient constructions*. In A. Juels, R. N. Wright, and S. Vimercati, editors, ACM CCS 06, pages 79–88, Alexandria, Virginia, USA, Oct. 30 – Nov. 3, 2006. ACM Press.
[26] R. Rivest, A. Shamir, and L. Adleman. *A method for obtaining digital signatures and public-key cryptosystems*. In Comm. of the ACM, 21:2, pages 120 ~ 126, 1978.
[27] R.L. Rivest, L. Adleman, and M.L. Dertouzos. *On data banks and privacy homomorphisms*. In Foundations of Secure Computation, 1978.
[28] Rosing, M. Implementing *Elliptic Curve Cryptography*. Greeenwich, CT: Manning Publications, 1999.
[29] S. Kamara and K. Lauter. *Cryptographic cloud storage.* In Proc. Workshop Real-Life Cryptographic Protocols and Standardization (RLCPS), pages 136{149, 2010.
[30] Safavi-Naini and Ran Canetti, editors, CRYPTO 2012, volume 7417 of Lecture Notes in Computer Science,
[31] T. Collins, D. Hopkins, S. Langford, and M. Sabin. Public Key Cryptographic Apparatus and Method. US Patent #5,848,159. Jan. 1997.
[32] T. El Gamal. *A public key cryptosystem and a signature scheme based on discrete logarithms*. In Advances in Cryptology, pp. 10-18. Springer Berlin Heidelberg, 1985.
[33] T. Takagi. *Fast RSA-type Cryptosystem Modulo pkq*. In H. Krawczyk, ed., Proceedings of Crypto 1998, vol. 1462 of LNCS, pp. 318–326. Springer-Verlag, Aug. 1998.
[34] Van Dijk, Marten, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. *Fully homomorphic encryption over the integers.* Advances in Cryptology EUROCRYPT 2010 (2010): 24-4
[35] Voltage Security, Inc. http://www.voltage.com/
[36] W. Diffie and M. Hellman, *Multiuser Cryptographic Techniques*." IEEE Transactions on Information Theory, November 1976
[37] www.nsa.gov , *The Case for Elliptic Curve Cryptography*
[38] Y.-C. Chang and M. Mitzenmacher. *Privacy preserving keyword searches on remote encrypted data*. In J.Ioannidis, A. Keromytis, and M. Yung, editors, ACNS 05,volume 3531 of LNCS, pages 442–455, New York, NY,USA, June 7–10, 2005. Springer, Berlin, Germany.
[39] Z. Brakerski and V. Vaikuntanathan, *Efficient Fully Homomorphic Encryption from (Standard) LWE*. Proceedings of FOCS 2011. Full version available at IACR eprint.
[40] Z. Brakerski and V. Vaikuntanathan, *Fully Homomorphic Encryption for Ring-LWE and Security for Key Dependent Messages*. In P. Rogaway (Ed.), CRYPTO 2011, LNCS, vol. 6841, Springer, 2011, pp. 505–524.
[41] Z. Brakerski, C. Gentry and V. Vaikuntanathan, *Fully Homomorphic Encryption without Bootstrapping*. Cryptology ePrint Archive, Report 2011/277
[42] Jean-Sebastien Coron, David Naccache, and Mehdi Tibouchi. *Public key compression and modulus switching for fully homomorphic encryption over the integers*. In David Pointcheval and Thomas Johansson, editors, EUROCRYPT 2012, volume 7237 of Lecture Notes in Computer Science, pages 446{464. Springer, 2012.
[43] Jung Hee Cheon, Jean-Sebastien Coron, Jinsu Kim, Moon Sung Lee, Tancrede Lepoint, Mehdi Tibouchi, and Aaram Yun. *Batch fully homomorphic encryption over the integers*. In Thomas Johansson and Phong Q. Nguyen, editors, EUROCRYPT 2013, volume 7881 of Lecture Notes in Computer Science, pages 315-335. Springer, 2013.
[44] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. *Can homomorphic encryption be practical?* In CSW, pages 113–124. ACM, 2011.
[45] P. Rogaway, *Evaluation of Some Block Cipher Modes of Operation.* Technical Report, Cryptography Research and Evaluation Committees (CRYPTREC), 2009.
[46] M. Nagendra and M. Chandra Sekhar, *Performance Improvement of Advanced Encryption Algorithm using Parallel Computation***.**
[47] www.porticor.com, *Securing Data in the Cloud*.
[48] N. P. Smart and F. Vercauteren, Fully Homomorphic Encryption with Relatively Small Key and Chiphertext Sizes, Public Key Cryptography (PKC) in Lecture Notes in Computer Science Volume 6056, 2010, pp 420-443.
[49] Depavath Harinath, "Enhancing Security through Steganography by using Wavelet Transformation and Encryption" in in International Journal of Modern Trends in Engineering and Research (IJMTER), vol.2, Issue8,July 2015

[50] Depavath Harinath et. al., "Cryptographic Methods and Performance Analysis of Data Encryption Algorithms in Network Security" in *International Journal of Advanced Research in Computer Science and Software Engineering* (IJARCSSE) ,Vol.5, Issue7, July 2015.

About Author (s):

Depavath Harinath, received Bachelor of Science degree in computerscience from New Noble Degree college, Affiliated to Osmania University, Hyderabad, Telangana, India in 2008 and received Master of Computer Applications degree from Sreenidhi Institute of Science and Technology, an autonomous institution approved by UGC. Affiliated to JNTU, Hyderabad, Telangana., India in 2012. Now working as Lecturer in Computer Science in HRD Degree and PG college, Affiliated to Osmania University, Narayanaguda, Hyderabad, Telangana, India. Having three years of experience in teaching and already published many manuscripts in different international journals. Research fields includes Computer Networks and Network Security.

Prof.M. V. Ramana Murthy, Professor in department of mathematics and computerscience, Osmania University, since 1985. Obtained Phd degree from Osmania University in 1985 and visited many a countries across the globe in various capacities and participated in many academic programs. Research fields includes computational plasma, Artificial Neural Networks, and Network securities.

Mrs B. Chithra, Asst .Professor in department of mathematics and computer science, Osmania University, Hyderabad. She has Worked as Associate Professor (dept. of Computer Science) at St. Ann's college for women, for about fifteen years and published a paper in one of the reputed International Journals. Research interest includes Network Security.

K.Ramesh Babu, received his Ph.D in 2012 from Osmania University for the thesis entitled " Estimation of target position from noisy radar data using kalman filter". Presently working at M.V.S.R Engineering College, Nadergul as Asst. Professor in mathematics. The research areas includes Theoretical computer Science.