Data Sharing towards Integrated Secure Key Encryption System with Big data in Cloud Computing

Sankara Rao
PG Scholar, Department of CSE,
BABA Institute of Technology and Sciences , AP

J Ratna Kumar
Assoc.Prof, Department of CSE,
BABA Institute of Technology and Sciences, AP

Abstract- These day's Cloud computing developed as a practical and demonstrated conveyance stage for giving business or purchaser IT services over the Internet. Be that as it may, the services given by the cloud are of outsiders, experiencing security and protection of the clients BIG Data is basic at cloud storage. This paper displayed a methodical report on the feasibility of elliptic bend cryptography in anchoring BIG Data at cloud storage. Based on the deliberate investigation a protected gathering correspondence structure is additionally displayed in this work. This work accept the cloud storage is raised with Hadoop based data focus. So as to do efficient investigation private key calculations (EC, RSA), open key calculations (AES, DES) and half and half calculation (Elliptic bend Dephihelmen ECDH) are considered to check the BIG Data security at cloud based data focus. From the trial results this examination prescribes that for the unbound key sharing channel Elliptic Curve (EC) based cryptographic calculations are very secure than RSA and private key calculations.
Keywords: Cloud Computing, Cryptography, Data Sharing, Encryption System, ABE, Cloud security, Encryption and Decryption Algorithm.

I. INTRODUCTION

Big data is a term that depicts the mind boggling or large volume of organized and unstructured data. It is hard to process utilizing available database the executive's instruments. To separate vital quality from Big Data, we require perfect preparing strategies, explanatory capacities and aptitudes. There are a few big data preparing methods including cloud condition accessible. In the latest two decades, the reliable augmentation of computational power has made an amazing stream of data. Huge data is ending up being progressively available and also increasingly reasonable to PCs. For example, the celebrated relational association, Facebook, serves 570 billion webpage visits for consistently, stores 3 billion new photos every month, and supervises 25 billion bits of substance. Google's request and notice business, Facebook, Flickr, YouTube, and Linkedin use a store of synthetic mental aptitude traps; require parsing boundless measures of data and settling on decisions quickly. Sight and sound data mining stages make it basic for everybody to achieve these destinations with the base proportion of effort to the extent programming, CPU and framework. All of these delineations exhibited that staggering huge data challenges and basic resources were administered to reinforce these data genuine activities which provoke high accumulating and data taking care of costs. The present advances, for instance, matrix and cloud computing have all normal to get to a ton of totaling in order to enroll compel resources and offering a lone structure see. Among these advances, cloud computing is transforming into a skilled development modeling to perform considerable scale what's increasingly, complex computing, and has vexed the manner in which that enlisting establishment is distracted and used. Likewise, a fundamental purpose of these advances is to pass on preparing as a response for taking care of gigantic data, for instance, large scale, multi-media and high dimensional data sets. Big data and cloud computing are both the speediest moving progressions recognized in Gartner Inc's. 2012 Hype Cycle for Emerging Technologies. Cloud computing [1] is

associated with new perspective for the acquirement of figuring establishment and gigantic data taking care of framework for an extensive variety of advantages. Likewise, some new cloudbased developments must be grasped because overseeing enormous data for concurrent getting ready is troublesome. Big data is an enormous volume of both organized and unstructured data that is so large to process with conventional database and programming strategies [2]. The meaning of big data as additionally given by the Gartner: "Big Data are highvolume, high-speed, as well as high-assortment data resources that require new types of preparing to empower improved basic leadership, understanding disclosure and process enhancement [3]. As per Wikimedia, "In data innovation, big data is an accumulation of data sets so large and complex that it ends up hard to process utilizing close by database the executives devices".

II. Related Work

As a default setting HADOOP accept secure system and no security structure is joined other than Kerberos based validation. At first, Park and Lee [1] present secure HADOOP design by joining the AES based encryption/decoding to HDFS engineering. As an expansion Das et al., [7] talked about the appropriation of Kerberos based confirmation instrument to anchor the data in HDFS storage. In comparative work Malley et al. [8] examined a Kerberos based verification system. For a client Kerberos produces a Token ID based on client ID, restored ID, and issue date and succession number. Created Token ID is shared to Namenode. Token validation is done at Namenode utilizing Master key which is chosen haphazardly by Namenode for check. Further confirmation is done at each Datanode utilizing the tokens created at Namenode based on square Id and comparing mystery key of the activity. Here the security is defenseless when the Datanode is endangered. Zhou and Wen [9] embraced Cipher Text Policy and Attribute Based Encryption (CP_ABE) to give get to control qualifications to cloud clients. Rather than utilizing clients individual identity CP-ABE utilizes property and a scrambled data get to control structure. The client can play out the decoding given by the match of client distinguishing proof attribute coordinate with access control structure. In this component the figure message and relating figure key created by CP-ABE strategy is sending to the Namenode. The Namenode further re-encodes the figure message and disseminates the record squares to Datanodes. Because of the incorporated key conveyance at Namenode based on CP-ABE, the key dissemination is by all accounts straightforward with less client intercession. In any case, security to the customer document isn't ensured as the first record is additionally sent to Namenode for re-encryption. Cohen and Acharya [10] portrays AES-NI based encryption structure for data encryption and honesty approval by making utilization of Trusted Platform Module (TPM). Progressed cryptographic instruments like homomorphism encryption additionally received to counter this issue. Jin et al., conceived a security component for cloud storage utilizing homomorphism encryption and client confirmation is completed utilizing operator innovation. Completely homomarphic encryption empowers different clients to work with scrambled contributions to create encoded results. In this way the trust can be ensured on the data storage. Then again specialist innovation empowers get to control instruments to get to the common assets. The real confinement is the completely homomarphic encryption isn't developed to adjust on certifiable application situations. A few cross breed encryption schemes likewise created to anchor data at HDFS. Lin et al.,proposed a Hybrid encryption strategy. Here clients' data record is symmetrically encoded by a remarkable key k and k is then unevenly scrambled with the proprietor's open key. This Hybrid encryption at first uses the DES calculation to produce the data key to scramble clients' files.

III. Key Encryption Schemes

A. Advanced Encryption Standard

NIST established the Advanced Encryption Standard (AES) in 2001 [1] 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 [6], 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 [6] 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 batch decrypt 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.[1, 3]

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. 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. 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.

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 [4] and improved constructions and security definitions are given in. 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 [4], achieved using a special two-layered encryption construct for each word that allows searching the ciphertexts with a sequential scan. Goh [5] 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 [5], Chang and Mitzenmacher [8] 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 pseudorandom 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 [1]. 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 [10] 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 minentropy. 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.

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 userprovided 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 [4]. 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 [6] 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 [2, 4] 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 [2, 3] 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 [1]. 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 [2]. 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 [6]. 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 [8]. 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 [8, 9]. 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 [4] 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 [3] 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 [2] 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 [3] 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 [1] 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."[4]. 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 KH. 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 $\mathbf{PK}_{FHE}$.

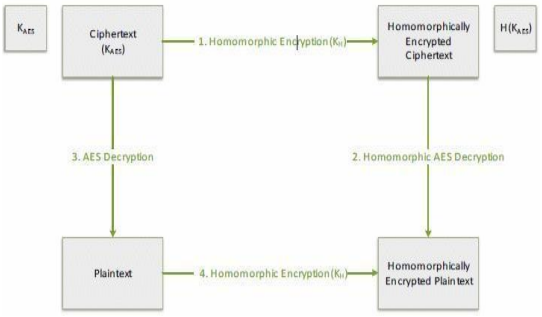

Figure 3. Homomorphic encryption using AES

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 [10] used a variant of Simple Fully Homomorphic Encryption scheme over the integers with the scale-invariant property as in [3]. 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.

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.

IV. Proposed Method

In the proposed system, new privacy protection scheme is introducing, that is the Ring signature. Ring signature is the closed group file authentication system, where the data owner in the group can able to securely share his files within the group without outsider's inference. And simultaneously integrity of the file is maintained. For each and every file the Ring Signature will be created. With the help of Ring Signature the group members can able to successfully decrypt and download the file from the Hadoop.

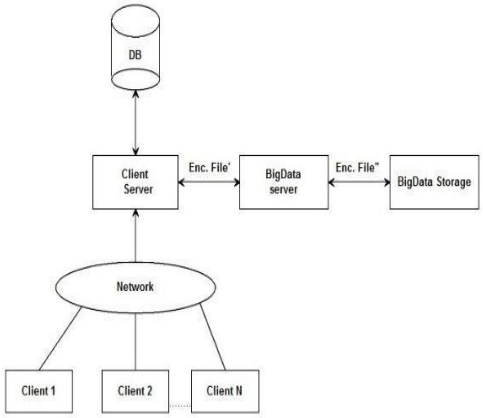

Fig -1: Proposed System Architecture

The above architecture explained about how we are using ring signature in order to provide security in big data. Admin will create a group by collecting their id. Then he will provide security keys to all users in the group. If client1 send the file, it will be stored in client server. In client server one database is maintained. After that file will be encrypted with the ring signature and it is send to big data server. File will be second time encrypted before going to the big data storage by using AES key. If anybody wants to download the file he has to upload the ring signature. Otherwise he cannot download the file from bigdata storage

Ring Signature Creation

User will get the public keys of all the members present in the user's group. Then he will generate the Hashcode for the uploading file. Using this, do XOR operation of the Hashcode with the public keys of all the members present in the user's group. The final result of XOR operation is the SecureMD (Secure Message Digest). Then Write the obtained SecureMD in to the file by that Secured file will be created. Then Encrypt the Secured file with the private key of uploading user, the Encrypted file is called Ring Signature of the uploading file. After creating Ring Signature user sends the Ring Signature file to the selected members of the group through the email.

File Encryption Process

- Fetch the AES Key

- Encrypt the uploaded file using AES Encryption Technique.

File Decryption Process

- Fetch the AES Key
- Decrypt the downloaded file using AES Decryption Technique

Ring Signature Verification Process

The destination user will get the public keys of all the members present in the user's group. Generate the Hashcode for the downloaded file. After that, do XOR operation of the Hashcode with the public keys of all the members present in the user's group. The final result of XOR operation is the SecureMD (Secure Message Digest), let us call it as SecureMD1. Decrypt the Ring Signature File with the public key of the uploaded user and get the SecureMD, let us call it as SecureMD2. Compare the SecureMD1 and SecureMD2. If matches then Ring Signature verification process successful and file will be successfully downloads to the Client system. Else Ring Signature verification process failure and it will give the error message that Ring Signature mismatching.

VI. Conclusion

This paper presents different ways to classify symmetric encryption algorithms. This analytical study gives a fair review on cryptographic techniques and pros and cons of various symmetric key encryption algorithms. This paper also shows the importance of block ciphers and block cipher modes of encryption and decryption to enhance security. This paper performed an analysis of various security attacks possible against block cipher encryption algorithms. Then different block cipher algorithms are compared based on their block size, key size, number of rounds, cryptographic structure, application areas and vulnerability against various security attacks. The performance evaluation of DES, 3DES, AES and Blowfish algorithms has done based on the throughputs of encryption and decryption. This analysis shows that AES is the better algorithm for Cloud, Bigdata, and AES does not have any well-known weak points so far. The AES algorithm with complex rounds, larger key size and larger number of rounds considered as more secure and resistant against all types of existing security attacks. Blowfish is considered as faster algorithm for small sized packets and it is suitable for IOT applications. This study shows the limitations of existing encryption algorithms in different perspective and relevance to modify it or to develop new algorithms to face challenges in a fast growing technological world.

References

[1] A Practical Public Key Encryption Scheme Based on Learning Parity with Noise: ZHIMIN YU1, CHONG-ZHI GAO, ZHENGJUN JING, BRIJ BHOOSHAN GUPTA, AND QIURU CAI

[2] A. Katal, M. Wazid, and R. H. Goudar, ``Big data: Issues, challenges, tools and good practices," in Proc. IEEE Int. Conf. Contemp. Comput., Aug. 2013, pp. 404_409.

[3] Z. Xiao and Y. Xiao, ``Security and privacy in cloud computing," IEEE Commun. Surveys Tuts., vol. 15, no. 2, pp. 843_859, May 2013.

[4] S. Singla and J. Singh, ``Cloud data security using authentication and encryption technique," Global J. Comput. Sci. Technol., vol. 13, no. 3,pp. 2232_2235, Jul. 2013.

[5] V. Goyal, O. Pandey, A. Sahai, and B.Waters, ``Attribute- based encryption for _ne-grained access control of encrypted data," in Proc. ACM Conf. Comput. Commun. Secur., Oct. 2006, pp. 89_98.

[6] J. Bethencourt, A. Sahai, and B. Waters, ``Ciphertext-policy attributebased encryption," in Proc. IEEE Int. Conf. Secur. Privacy, May 2007, pp. 321_334.

[7] K. Yang, X. Jia, and K. Ren, ``Secure and verifiable policy update outsourcing for big data access control in the cloud,'' IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 12, pp. 3461_3470, Dec. 2015.

[8] X. Boyen and B.Waters, ``Anonymous hierarchical identity-based encryption (without random oracles)," in Proc. Adv. Cryptol. (ASIACRYPT), vol. 4117. Aug. 2006, pp. 290_307.

[9] M. Green and G. Ateniese, ``Identity-based proxy re- encryption," in Proc. Int. Conf. Appl. Cryptogr. Netw. Secur., 2007, vol. 4521. pp. 288_306.

[10] C. Gentry, ``A fully homomorphic encryption scheme," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2009.