iJRASET
International Journal For Research in
Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 Issue: III Month of publication: March 2018

DOI: http://doi.org/10.22214/ijraset.2018.3235

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue III, March 2018- Available at www.ijraset.com

Securing Cloud Commodity Data Through STOE Algorithm

Mrs. K. Priya[1], Mr. Raveendiran[2]
[1, 2] Department of Computer Science And Engineering, Dhaanish Ahmed College Of Engineering, Padappai, Chennai-601 301.

Abstract: A secure method that could prevent most attacks from both insiders and outsiders of networks to break data confidentiality, and meanwhile with constant communication cost for re-encrytion anti incremental scale of process. Here proxy re-encryption scheme is used to provide data security in Cloud-Assisted process.With the agile growth of the amount of information, cloud computing servers need to process and analyze large amounts of unstructured data so it is necessary to provide security to the data that is stored in commodity server. The proposed system provides security to data that is stored in commodity server using STOE algorithm and Mincounter technique where we can further improve data integrity and data security while communicating and storing the data in commodity server for better performance.

Keywords: Data confidentiality, data security, commodity server, Mincounter technique, STOE Algorithm.

I. INTRODUCTION

In the epoch of Big Data, a, cloud computing servers need to process, store and analyze large amounts of data timely and accurately. Enormous amount of data require idealistic large storage and high-performance computation that a normal user requires. Cloud is popularly applied to leverage the computation and storage capability of a cloud for data . A cloud is a powerful platform that can provide additional conveniences as a data distribution agent. When a user has forensic requests for certain data being collected, stored and accessed, he can directly authorize the requests to the cloud at any time with greater convenience. Due to space inefficiency and high-complexity of data, commodity server storage is used in proposed system. The disadvantage is providing security to data that is stored in commodity server . A framework based on cryptographic methods is used to support data security in cloud and commodity server . The various existing methods that are used for providing data security is CIBPRE , ECC and Mincounter technique. The proxy re-encryption scheme named conditional identity-based broadcast proxy re-encryption (CIBPRE) is used for better performance because it allows a sender to encrypt a message to multiple receivers by specifying these receivers' identities, and the sender can delegate a re-encryption key to a proxy so that he can convert the initial ciphertext into a new one to a new set of intended receivers. The Elliptic-curve cryptography (ECC) is used to prevent attacks from both insiders and outsiders to break data confidentiality using keys. ECC is a public key encryption technique based on *elliptic curve theory* that can be used to create faster, smaller, and more efficient cryptographic keys. ECC generates keys through the properties of the elliptic curve equation instead of the traditional method of generation as the product of very large prime numbers.

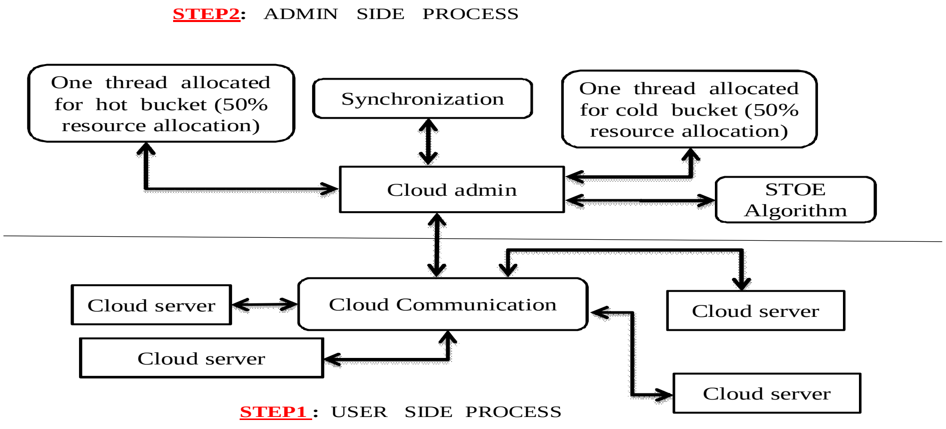

STEP2: ADMIN SIDE PROCESS
One thread allocated for hot bucket (50% resource allocation)
Synchronization
One thread allocated for cold bucket (50% resource allocation)
Cloud admin
STOE Algorithm
Cloud server
Cloud Communication
Cloud server
Cloud server
Cloud server
STEP1 : USER SIDE PROCESS

Fig1: Architecture Diagram

When the data is transferred from cloud server to commodity server, data integrity is maintained through Mincounter technique. MinCounter is a cost-efficient cuckoo hashing scheme. The idea behind MinCounter is to reduce the occurrence of endless loops in the data insertion by selecting unbusy kicking-out routes. MinCounter selects the "cold", rather than random, buckets and avoids "hot" paths to alleviate the occurrence of endless loops in the data insertion process when hash collisions occur. The salient features of mincounter is offering efficient insertion and query services and delivering high performance of cloud servers, as well as enhancing the experiences for cloud users. MinCounter takes advantage of "cold" buckets to alleviate hash collisions and decrease insertion latency. MinCounter optimizes the performance for cloud servers, and enhances the quality of experience for cloud user.

II. LITERATURE SURVEY

In 2018 Wei Wang et al. proposed "Secure Data Collection, Storage and Access in Cloud-Assisted IoT". They provides to a promising solution to data booming problems for the ability constraints of individual objects. The main advantage of cloud-assisted IoT data managing method to keep data confidentiality when collecting, storing and accessing IoT data with the assistance of a cloud with the consideration of user's increment. The proposed method helps to keep secure data on cloud for storing and accessing.

In 2006 Yuanyuan Sun et al. proposed "A Collision-Mitigation Cuckoo Hashing Scheme for Large-scale Storage Systems". They proposed MinCounter technique to alleviate the occurrence of endless loops in the data insertion by selecting un busy kicking-out routes. The main advantage of MinCounter is it selects the "cold" (infrequently accessed), rather than random, buckets to handle hash collisions. The proposed method of salient features offering efficient insertion and query services and delivering high performance of cloud servers, as well as enhancing the experiences for cloud users.

In 2013 Rachna Arora, Anshu Parashar et al. proposed "Secure User Data in Cloud Computing Using Encryption Algorithms". They proposed Cloud Computing which solves many problems of conventional computing, including handling peak loads, installing software updates, and using excess computing cycles. The main advantages has also created new challenges such as data security, data ownership and trans-code data storage.

In 2013 Zuzana Priscaková et al. proposed "Model Of Solutions For Data Security In Cloud Computing". They proposed a model which is used to ensure data stored in the cloud. The main advantages of this model is, it is divided into 7 modules. Each module is apparent from the terms of data security and described specific situations when working with data. The proposed method is to convert the implementation of cloud into enterprise environments with respect to data security in the firm..

In 2013 Maha TEBAA et al. proposed "Secure Cloud Computing through Homomorphic Encryption". They proposed a cloud computing process .A number of benefits and services to its customers who pay the use of hardware and software resources (servers hosted in data centers, applications, software...) on demand which they can access via internet without the need of expensive computers or a large storage system capacity and without paying any equipment maintenance fees. The main advantage of The proposed method helps in the cloud providers must provide guarantees on the protection of privacy and sensitive data stored in their data centers shared between multiple clients using the concept of virtualization.

III. IMPLEMENTATION

The following are the modules which consists of implemention work and working principles.

A. Cloud Server Creation

A cloud server is a logical server that is built, hosted and dispatched through a cloud computing platform over the Network. Cloud servers possess and exhibit similar capabilities and functionality to a typical server but are accessed remotely from a cloud service provider. A cloud server also be called as virtual server or virtual private sever. After the creation of virtual server it allows the user to store and retrieve data from anywhere anytime. In cloud server the space will be allocated by admin and a threshold limit will be set by which we will find the maximum limit of user's data storage.

Through this we can set an alert when the storage space reaches 80% of total space. If this alert arises then the cloud server should choose commodity server to store further datas of the user. So only the commodity server will be analysed and connected at the beginning itself. Cloud servers are created and it will handle and analyze large amounts of high-dimensional and unstructured data timely and accurately. Queries will be processed in cloud servers.

B. Dividing hot bucket and Cold Bucket

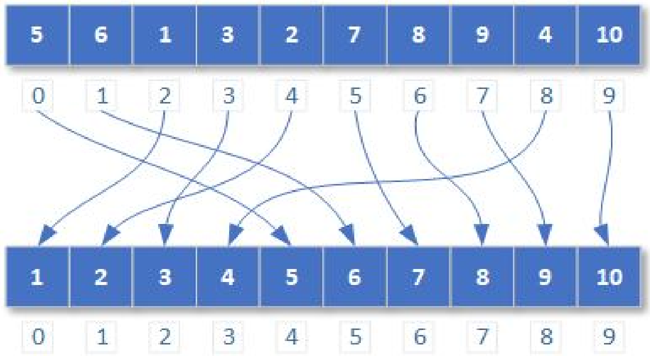
Hot bucket which means frequently accessed which will be under one process and another the cold bucket which is infrequently accessed will be under another process. Both are divided according to the duration and accessing time. The following steps are involved in this module:

1) *Step1:* Start the process of transferring the data from cloud to commodity server.
2) *Step2:* First take 100mb of data that is to be verified.
3) *Step3:* Initially this data enters into hot bucket.
4) *Step4:* From this hot bucket, cold bucket fetches 10mb of data for verifying.
5) *Step5:* After finishing all the process, data is send to commodity server for storage.

C. Dual Threaded Selection Algorithm

Dual threaded selection algorithm is used to simultaneously handle both hot bucket and cold bucket which can be done by invoking two threads. These threads will provide equal importance and complete the task efficiently. Even though if hot bucket locks under infinite loops or the total reconstruction occurs the cold bucket will help to recover.

A sorting algorithm (defined) - an algorithm that arranges all data items of an array in a specific numerical or lexicographical logical order (see figure below). The sorting algorithms are typically used to optimize the complexity and improve the performance of other algorithms, as well as to make the sorted data canonical producing human-readable outputs. In fact, sorting algorithms have lots of known applications in the field of commercial data processing, as well as the modern scientific computing in general.



Sorting Array of Data Items

The most common applications of various sorting algorithms basically include financial transactions processing, mathematical and economical statistics, linear algebra and functional optimization, data compression and visualization, cryptography, linguistic systems, timeline and tasks scheduling, logs and reports, genetic programming, AI data mining and neural networks, etc. In fact, there's the number of reasons why actually we might want to use sorting as a part of the other more complex algorithms. For example, the solution of the trivial statistical sub-problem of finding top N-students with the highest score, can be simplified by using a sorting algorithm. Also, sorting can be used in various data compression algorithms to find those sequences of characters that more or less frequently occur in the text message being encoded. In genetic programming we use sorting algorithms to obtain chromosomes in the current population that represent the fittest problem's solution candidates. Moreover, the using of sorting algorithms allows to optimize the process of retrieving the most up-to-date information from a database such as the list of the latest remote transactions to a server ordered by date and time. From the beginning of the era of computing, there was the number of various trivial sorting algorithms formulated. These algorithms basically include the famous bubble sort, as well as the selection and insertion sorting algorithms that are mainly based on performing the exchange-based stable data sorting. Also, there's a special kind of fast and efficient methods of sorting such as quicksort, merge and heapsort algorithms, capable to sort datasets with a typically huge amount of items drastically faster compared to those elementary methods of sorting mentioned above. From one respect, many of those algorithms provide an efficient method for sorting arrays of data items having the either fundamental or abstract data type (ADT). From the other respect the using of each of those algorithms might become inefficient due to the number of potential drawbacks persistent in those algorithms, having an impact on the general performance of the process of sorting arrays with typically huge amount of data items.

The process of sorting itself still remains one of the most heavy tasks performed by modern computers, normally consuming large amounts of the processor's and system memory resources. That's why, the sorting of enormously huge amounts of data items might become impossible due to performance degrades caused solely by the nature and specific of a sorting algorithm applied. Specifically, the main purpose of the following article is to introduce a new fast sorting algorithm that provides more efficiency and performance speed-up compared to some of those known implementations of the existing sorting algorithms such as either a classical quick and heap sort. The new fast sorting algorithm being discussed is mainly based on an improved introspective sort (introsort) that employs more than one sorting algorithm such as either the most efficient 3-way quicksort or the trivial classical insertion sort to perform the actual sorting. Rather than the conventional introspective sort, proposed by David Musser in 1997, the following algorithm entirely relies on the observation of the input array, determining its size and initial order to properly select an optimal backend algorithm (e.g. 3-way quicksort vs. insertion sort) to be used for the actual sorting. Further, in this article, we'll thoroughly discuss about both of these sorting algorithms in details. In addition, we'll also discuss particular aspects of the other algorithms such as an adjacent and cocktail shaker sort used to perform an auxiliary and small arrays pre-sorting. Furthermore, since the algorithm being discussed has been formulated, we'll spend just a bit more time to the number of effective algorithm-level performance optimizations that allow to drastically increase the performance speed-up of the process of sorting, making the algorithm being introduced even much more (2x - 6x times) faster than the number of known implementations based on the quicksort and heapsort fast algorithms, that, in turn, were formulated as a part of specific industrial standards for C and C++ programming languages.

D. Dead lock avoidance Through Synchronization

When two threads are running then there may be deadlock situation occurs which will be avoided through the synchronization technique. This will greatly help to avoid deadlock situation and becomes fault tolerant. It can be performed by the process of BANKERS algorithm. Bankers's Algorithm is resource allocation and deadlock avoidance algorithm which test all the request made by processes for resources, it check for safe state, if after granting request system remains in the safe state it allows the request and if their is no safe state it don't allow the request made by the process.

Inputs to Banker's Algorithm:
1. Max need of resources by each process.
2. Currently allocated resources by each process.
3. Max free available resources in the system.

Request will only be granted under below condition.
1. If request made by process is less than equal to max need to that process.

2. If request made by process is less than equal to freely availbale resource in the system.

E. STOE Algorithm

The new algorithm called STOE algorithm is created to process on the stored data to provide security. Each and every cloud servers will store data in commodity server by following "Security Through Object Encryption (STOE)" Algorithm.

When the cloud server reaches 80% of storage then the future incoming data of clients are stored in commodity server automatically. The data of clients are stored as a key in commodity server and as metadata in cloud server. This key is generated using ECC method and STOE algorithm. Here security is provided through object encryption. Hence, this algorithm is called STOE(Security Through Object Encryption) algorithm.

Thereby the original owner of the commodity server is unable to view the stored data and the details of the clients who uses the commodity server for storage.

Through this algorithm, the data will be more secured in commodity server. By using this algorithm the cloud server has assured security for the data stored by their clients. The security for cloud server will be provided by itself . So this security will act as second guard for the stored data in commodity server. Finally we are providing dual security to our client's data that is stored in commodity server.

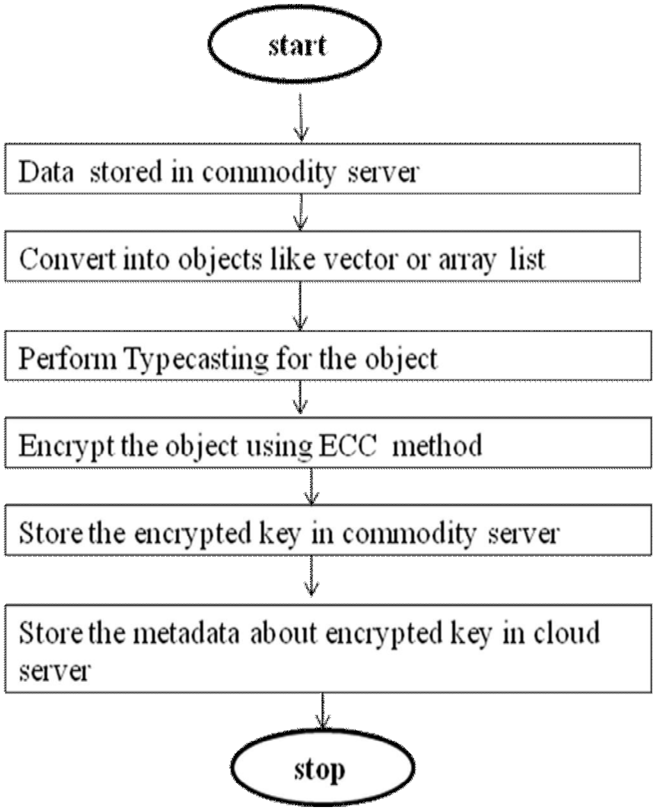

start
Data stored in commodity server
Convert into objects like vector or array list
Perform Typecasting for the object
Encrypt the object using ECC method
Store the encrypted key in commodity server
Store the metadata about encrypted key in cloud server
stop

Fig2: STOE algorithm working

Data integrity is maintained between cloud server and commodity **server** using Mincounter technique. All the metadata about encrypted data is stored in cloud to provide better security.

IV. ALGORITHM DESCRIPTION

A. Advanced Cuckoo Hashing Technique

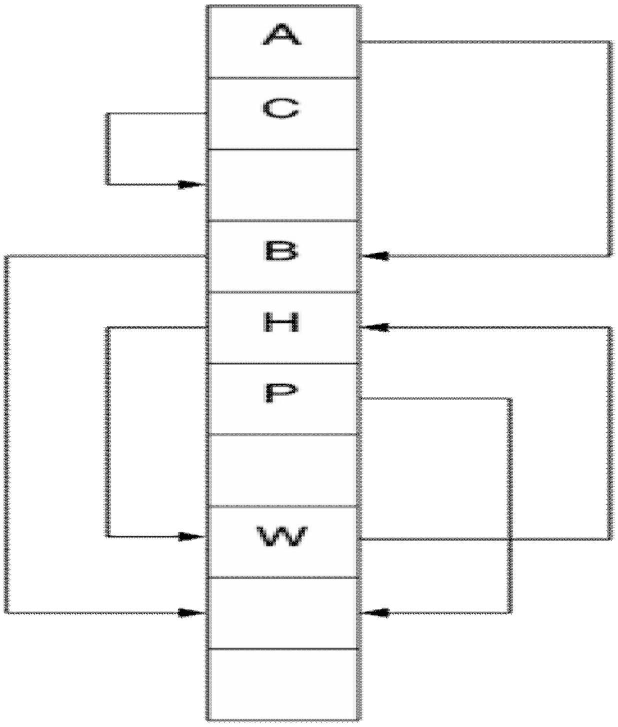
Standard Cuckoo Hashing The cuckoo hashing is a dynamization of a static dictionary and supports fast queries with the worst-case constant-scale lookup time due to flat addressing for an item among multiple choices.

1) *Definition 1:* Standard Cuckoo Hashing. Conventional cuckoo hashing uses two hash tables, T1 and T2, with the length m, and two hash functions h1, h2:U →{0,...,m−1}. Each item x∈S is kept in one of the two buckets: h1(x) of T1, h2(x) of T2, but never in both. The hash functions hi, i = 1,2, meet the conditions of independent and random distribution.

We use arrows to show possible destinations for moving items as shown in Figure . If item x is inserted into hash tables, we first check whether there exists any empty bucket of two candidates of item x. If not, we randomly choose one from candidates and kick out the original item. The kicked-out item is inserted into Table2 in the same way. The process is executed in an iterative manner, until all items find their buckets. Figure demonstrates the running process that the item x is successfully inserted into the Table1 by moving items a and b from one table to the other. While, as shown in Figure ,endless loops may occur and some items fail to find a suitable bucket to be stored. Therefore, a threshold "Max Loop" is necessary to specify the number of iterations.

If the iteration times are equal to the pre-defined threshold, we can argue the occurrence of endless loops, which causes the re-construction of entire structure. Moreover, due to essential property of random choice in hash functions, hash collisions cannot be fully avoided, but significantly alleviated . It is shown in that if two hash tables are almost half full, i.e., $m \geq (1+ \varepsilon)n$ for a constant parameter $\varepsilon > 0$, and h1 and h2 are chosen uniformly and randomly from an (O(1),O(logn)) - universal family, the probability of being unable to arrange all items of dataset S is O(1/n), according to h1 and h2 Improved Cuckoo Hashing There is an improvement in cuckoo hashing and allows each item to own d > 2 candidate locations, which has been widely used in real-world applications [32], [36], [37], [40]–[44].

2) *Definition 2:* Improved Cuckoo Hashing. Improved cuckoo hashing leverages d hash tables, T1,T2,...,Td, and d hash functions h1,h2,...,hd : U →{0,...,m−1}, where m is the length of each hash table. Each item x∈S is kept in one of the d buckets: h1(x) of T1, h2(x) of T2, ..., hd(x) of Td, but never in d buckets. d(> 2) is a small constant.



Concurrency Control Using Locking It's important to efficiently support concurrent access to a cuckoo hash table. Some previously proposed schemes are used to improve concurrency. In order to support arbitration for concurrent access in data structures, locking is an efficient mechanism. Multi-thread applications achieve high performance through taking advantage of more and more cores. In order to ensure thread correctness and safety, a critical section controlled by a lock is used to allow the operations of multiple threads to be serialized when accessing the shared part of data.

B. *Coarse-Grained Lock.*

A simple way of locking is to attach a coarse-grained lock to the entire shared data structure, and only one thread can possess the lock in the meanwhile. The thread with the lock prevents other threads from accessing the shared data, even through others only want to read the shared data and have no updates, which has negative influence on concurrent performance.

C. *Fine-grained Lock.*

Another locking is to divide the coarse-grained lock into multiple fine-grained locks. Each fine-grained lock protects only one part of the shared data structures. Hence, multiple threads can work on different parts of the structures without conflicts. Fine-grained locking has positive influence on concurrent performance compared with the coarse-grained locking. However, careful design and implementation are needed to avoid deadlock, livelock, starvation, etc.
It is challenging to effectively support concurrent access to cuckoo hash tables. In order to support the concurrent execution of the single-thread cuckoo hashing algorithm, while still maintaining the high space efficiency of cuckoo hashing, we need to deal with two problems.

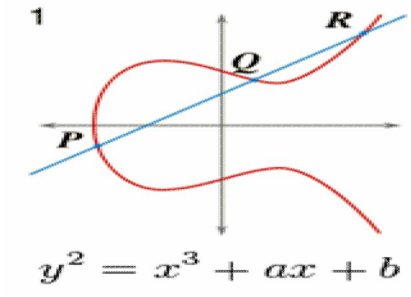
V. ELLIPTIC-CURVE CRYPTOGRAPHY

The equation of an elliptic curve is given as,

$$y^2 = x^3 + ax + b$$

Few terms that will be used,

E -> Elliptic Curve
P -> Point on the curve
n -> Maximum limit (This should be a prime number)



The fig 3 shows simple elliptic curve.

A. Key Generation

Key generation is an important part where we have to generate both public key and private key. The sender will be encrypting the message with receiver's public key and the receiver will decrypt its private key.
Now, we have to select a number **'d'** within the range of **'n'**.

Using the following equation we can generate the public key

Q = d * P
d = The random number that we have selected within the range of (1 to n-1). P is the point on the curve.
'Q' is the public key and 'd' is the private key.

B. Encryption

Let 'm' be the message that we are sending. We have to represent this message on the curve. This have in-depth implementation details. All the advance research on ECC is done by a company called certicom. Conside *'m'* has the point *'M'* on the curve *'E'*. Randomly select 'k' from [1 – (n-1)]. Two cipher texts will be generated let it be C1 and C2.

C1 = k*P,C2 = M + k*Q

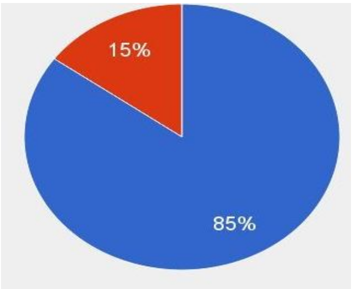
C1 and C2 will be send.

C. Decryption

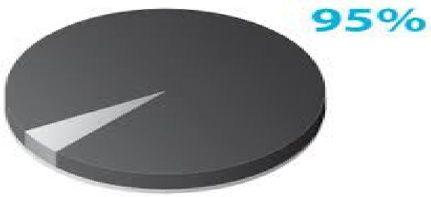
We have to get back the message 'm' that was send to us,
M = C2 – d * C1
M is the original message that we have send.

= M (Original Message)

VI. RESULT ANALYSIS



This figure represents 85% of security and 15% of chances of occurring attacks in commodity server before using STOE algorithm. Due to this data integrity gets affected which reduces the performance of commodity server.



This figure represents 95% of security that is provided to commodity server using STOE algorithm. Due to this security, data integrity is maintained and performance is increased.

VII. CONCLUSION AND FUTURE ENHANCEMENT

Cloud is a popular and useful framework for handling massive data. This paper focuses on providing security to commodity server when the data is collected, stored and accessed and introduces a new algorithm called "STOE" algorithm for this purpose. MinCounter [6] optimizes the performance for cloud servers, and enhances the quality of experience for cloud users.

Data stored in commodity server is more secured than before and all its metadata is stored in cloud server. So that the details about stored data in commodity server is invisible to others. Due to this security algorithm we can prevent leakage of confidential data stored in commodity servers. Here ECC technique is used where encryption is easy but decryption is hard because of this property the key stored in commodity server is hard to decode for attackers to break the data confidentiality.

The future enhancement is developing different types of security algorithms for different types of commodity server which increases the security of commodity server. Thereby the administrator can use many commodity servers with low budget and high security.

REFERENCES

[1] Wei Wang, Peng Xu and Laurence Tianruo Yang,"Secure Data Collection, Storage and Access in Cloud-Assisted IoT" , 2018.

[2] Y. Sun, Y. Hua, D. Feng, L. Yang, P. Zuo, and S. Cao, "Mincounter: An efficient cuckoo hashing scheme for cloud storage systems,"Proc.MSST, pp. 1–7, 2015.

[3] Rachna Arora, Anshu Parashar "Secure User Data in Cloud Computing Using Encryption Algorithms "Vol. 3, Issue 4, Jul-Aug 2013.

[4] Zuzana Priščáková and Ivana Rábová "MODEL OF SOLUTIONS FOR DATA SECURITY IN CLOUD COMPUTING "International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), Vol.3, No.3,June 2013.

[5] Maha TEBAA, Said EL HAJII "Secure Cloud Computing through Homomorphic Encryption" - International Journal of Advancements in Computing Technology(IJACT) Volume5, Number16, December 2013

[6] P. Xu, T. Jiao, Q. Wu, W. Wang, and H. Jin. "Conditional Identity-based Broadcast Proxy Re-Encryption and Its Application to Cloud Email" [J]. IEEE Transactions on Computers, 65(1), pp. 66-79, 2016.

[7] Yuanyuan Sun ; Yu Hua ; Dan Feng ; Ling Yang ; Pengfei Zuo ; Shunde Cao ; Yuncheng Guo, "A Collision-Mitigation Cuckoo Hashing Scheme for Large-scale Storage Systems" 2016.

[8] David Eppstein, Michael T. Goodrich, Michael Mitzenmacher, Paweł Pszona "Wear Minimization for Cuckoo Hashing: How Not to Throw a Lot of Eggs into One Basket",2014

[9] Hong Jiang, Dan Feng, Yu Hua, "FAST: Near Real-Time Searchable Data Analytics for the Cloud",2015

[10] Qiuyu Li , Yu Hua , Wenbo He , Dan Feng , Zhenhua Nie , Yuanyuan Sun , "Necklace: An efficient cuckoo hashing scheme for cloud storage services", 2014.

[11] https://bithin.wordpress.com/2012/02/22/simple-explanation-for-elliptic-curve-cryptography-ecc/(ECCALGORITHM)