Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

A Scalable Data Partitioning Framework to Enhance Cloud Security

Aditya Lal
Department of Computer Engineering
Georgia Institute of Technology (Georgia Tech)
Georgia, Atlanta, USA

***Abstract* — The huge data on the websites due to rapid use of cloud computing increase the security concern. Cloud computing provides various services such as storage, scalable transactions, sharing and downloading. There are many techniques which provides security of these services but these are dynamic in nature and leads to overload the memory. In this paper, a novel framework has been presented based on data partitioning technique which avoids the data duplication and provides scalable consistent data with enhanced security. The proposed work divided into three parts in which Dice Coefficient and Cosine Similarity has been used to secure the similar data. The data has been further encrypted using the encryption system to enhance the data integrity and security level. The encrypted data further consider as input for the data partition techniques in which different algorithms has been developed to partition the data in blocks. The proposed work further optimized using the Artificial Bee Colony algorithm to optimize the partitioned data. The obtained results further cross validated using the classification technique Artificial Neural Network (ANN) in which data has been trained and tested to check the effectiveness of the proposed work. The experimental results finally validated by measuring the True Positive Value (T_P), False Negative Value (F_N), True Negative Value, and False Positive value. The good T_P and bad F_N shows that proposed framework efficiently partitioned the data in an effective and scalable way.**

Keywords— Data Partitioning, Data integrity, Similarity Measures, ABC, ANN

I. INTRODUCTION

Data partitioning is a vital concept in the field of data processing system which provides the management of data in a scalable and consistent way. There is a large accumulation of data in a cloud environment. Cloud computing provides diverse range of services such as storage resources and computing models in a cost-effective manner. These services attract the users to use diverse range of cloud services which ultimately generates thousands of data on internet. Therefore, accumulation of such large amount of data requires development of data management techniques to manage bulk of data (Khedkar 2014). These methods plays a vital role in maintaining the consistency and scalable transactions. However, conventional data management methods inefficient to scale out due to low cost hardware such as servers. Therefore, researchers consider the SQL data stores to manage such information in a concise way. The various properties such as scalability, elasticity, and availability are the key values. However, scalability has been acquired using the data partitioning techniques scalability (Ahirrao and Ingle, 2015). It is a common method used to perform the scaling operation.

In a cloud environment, when the user requests the information, and if it is not present on one partition then it is completed by other partition. Such behavior in the cloud has been easily tracked and patterns are easily identified. Such type of patterns also called Data Access Pattern. Consequently, partitioning can be static or dynamic. In static partitioning, data items which are related to each other has been put together in one partition. The formed partitions cannot be modified and altered, once these are developed. Other partitioning techniques include hash, range partitioning which are easily access, but distributed nature often seen when data accessed from the servers. Although, existing partitioning methods and algorithms has been developed to manage bulk data but sometimes these algorithms do not perform their functioning well in case when the access patterns for data changes. Thus, there is a requirement to implement a partitioning scheme based on data management in cloud by ameliorating. An efficient and reliable partitioning scheme access minimum partitions when the user requirement to access the data from the server increases. However, partitioning system include assignment of servers to only authorized users. The assignments of m data partition system among K servers is m^K. The different servers represented by j among K severs is $\binom{K}{j}$. For group of J servers, the possible assignments in one partition is m data partition system in which specific servers p in one partition having no location assigned represented by $(J-p)^m$ can acquire combination of servers with at least one partition. The assignment to the server in a partition system only possible through the exclusion and inclusion principle (Levitin et al., 2017).

$$\sum_{p=0}^{J-1}(-1)^p\binom{J}{p}(J-p)^m \qquad (1)$$

The probability of assigning severs in one partition to the host with m partition system is given as

$$r(m,k,J)=m^{-K}\binom{K}{j}\sum_{p=0}^{J-1}(-1)^p\binom{J}{p}(J-p)^m \qquad (2)$$

Let us consider that fixed J out of m $(1\leq J\leq \min(m,k))$ servers has been assigned to host in a particular partition. To enhance the security in a partition, the attackers represented by q among k servers is q^K. The total number of assignments in which attackers no more distributed in a particular partition such as $m-p$ servers are $m-p^K$. When a fixed servers assigned in one partition then number of assignments in which attackers and host does not co-reside in p specific servers host in a specific partition is $m - p^K$. The exclusion and inclusion principle has been used in such scenario in which attackers and other hosts does not coincide given as

$$\sum_{p=0}^{J-1} (-1)^p \binom{J}{p} (m-p)^K \quad (3)$$

Thus, overall probability that specific servers assigned in a particular partition of J servers hosted with least number of attackers are

$$i(m,K,J) = \begin{cases} m^{-K} \sum_{p=0}^{J-1} (-1)^p \binom{J}{p} (m-p)^K, & K \geq J \\ 0, & K < J. \end{cases} \quad (4)$$

Fig. 1 represents the probability of attackers in a particular partition system where w represents the attackers successfully attacked in a particular area to access the server in a cloud. The assignment of m partition system with K servers and hosts J. When m=5, then attacked probability increases slowly in which 25 to 35 servers assigned having bulk data divided into five times. Consequently, when m =10, then probability of attackers rises thus, it again surges slowly when 50 to around 60 servers assigned and attacker probability increases as the servers and the hosts surges. Thus, need to partition the data also surges.

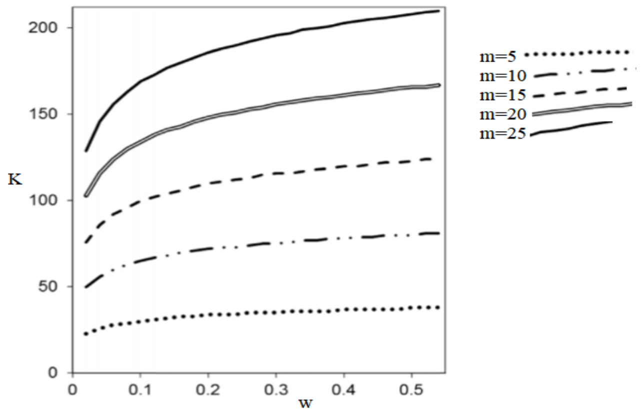

Fig. 1. Probability of minimum servers hosted in the data partitioning system as a function of m and w.

In this paper, the proposed work includes the efficient designing of storage system which ensures data availability and its correctness in a cloud using the partitioning algorithms. However, partitioning can be done in horizontal or vertical directions to control and manage the data. In addition, security criteria also utilized to prevent the system from intruders. The storage and data retrieval process can be simplified by lessen the data storage space and in the time of data requirement, data can be retrieved by merging data partitioning algorithms which provides correctness. Consequently, the integrity of cloud storage performed by comparing and equalizing the digital data signatures. The digital signature stored in third party auditor (TPA) has been extracted before the data transmission over the server. Furthermore, when the data retrieval process taken place, then extracted data has been compared with the data of digital signature stored in third party auditor. When comparing, if both data has been same, then integrity has been maintained.

Fig. 2 depicts the network of Cloud computing, which a large network spread over the internet. The cloud service provides directly linked with the TPA to securely transfer the message. TPA further communicates with different users to send the message request. The data also accessed through the mobile phones, and laptop by ensuring that reliable internet connection must be established to communicate. In addition, there is also direct communication between users and service providers.

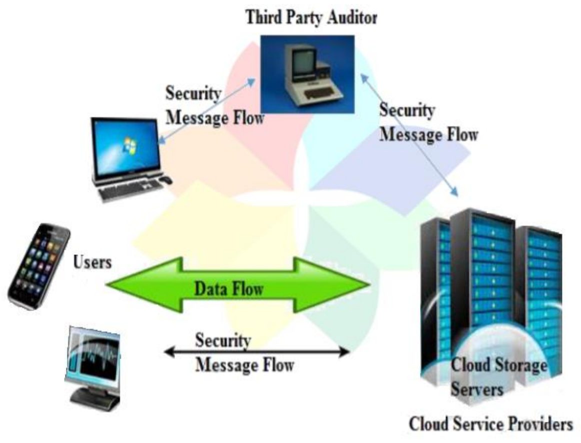

Fig. 2. Cloud Computing & Storage

Furthermore, there are wide range of cloud services provided in terms of large resources and good data storage which attract the users. But, there is a major issue of security which needs to be mitigated. Therefore, this paper introduces a novel data partitioning technique to secure and handling the bulk data using the data partitioning algorithms. The cosine similarity and dice coefficient has been used to store the similar data in one partition and other data in the other partitions. The data stored in the partitions further optimized using optimization algorithm Artificial Bee Colony (ABC). The scalable data further cross validated using the classification algorithm.

II. RELATED WORK

This section lights the data partitioning techniques used by the researchers and practitioner to solve the complex problems such as data storage, scalable transactions, and it's processing. The conventional data storage methods entirely relational in which large database has to be maintained to accumulate the bulk data by partitioning. Therefore, researchers utilize the computational techniques to store the big data. A data model has been developed which is further investigated in a customized environment. They also define the different problems to calculate the multi-frame data. The partition methods and data streaming models provides a way to compute the limitations of a single frame data. The face recognition system has been finally used to determine the effectiveness of the developed technique (Li et al., 2017). The biometric system and data partitioning techniques has been used to enhance the security level. The large accumulation of data on cloud being a worried concern when considering the security perspective. The use of TPA ensures to increase the security level. In addition, one time password system has been incorporated to further improve the security level. However, process of partitioning establish at the TPA side where files have been fragmented into small blocks and these are encrypted with secure code using cryptographic techniques. But the main cons of such method is searching mechanisms in accessing the encrypted files still not provided which restricts its usage (Shinde, 2015). Therefore, practitioners combine the data partitioning and encryption techniques together to revamp the cloud security. The designed mechanism divides the data received from the user and the separated files further encrypted which is further send to the server. When data is required back from the cloud then user simply request the data and decrypt it. The obtained data further combined so that whole file received by the user. The proposed structure is flexible, simple and reliable. But, missing of searching mechanism in a designed structure reduces the performance (Kanade et al. 2015). Consequently, scholars understand the importance of data partition techniques to improve the cloud storage. Therefore, seed block algorithm has used to efficiently recover the data. The security provided by the data encryption and decryption techniques for integrity. The large files divided into small parts to simplify the process and further send to the server using the TPA. But there is a possibility of losing file when the server is down so Seeback algorithm avoids such a problem. However, outsourced computation seems to be major barrier in enhancing the security level (Shaikh et al. 2015). So, three parameters such as Key size, tuple size and data formats has been consider to partition the data. The data partition strategies has been studied to develop methods which are broader and realistic in establishing the partitioning methods. The proposed work also includes the partition benchmark which help the other researchers in future to evaluate the partition methods (Zhang et al., 2019). But these methods are distributive in nature which also allow the unauthorized users to access the data. This is due to the distribution of virtual machines in an environment where authorized and unauthorized user's persists. Therefore, optimal partitioning policy has been used where optimal user's use the virtual machines to fix the attacker's. The proposed model effectively mitigates the attacks with minimum loses in a cloud environment using partitioning algorithms. However, these algorithms also used to revamp the scalability. But the main drawback of assigning machines using the data partition system is that attackers may access the machines during the assignment procedure which hampers its performance (Levitin et al., 2017). Thus, to avoid such problem a theft balancing technique developed to eliminate the co-resident attacks probability. A data protection procedures has been followed using partitioning algorithms where sensitive data reside in one partition and malicious data in other partition. If the separated blocks corrupted by the attackers, then they may have access to all meaningful information. Thus, optimal blocks has been acquired, which reduces the data access probability by the attackers. The effectiveness of the developed model checked considering the game theory model. But, most harmful behaviour of the attackers still needs to differentiate which puts a constraint in partition the blocks (Xeng, 2017). Therefore, scholars consider the data mining techniques using the horizontal partitioning schemes to avoid the accidental data protection. There is a minimum risk of data privacy using the association rules against the attackers. Furthermore, security procedure has been analysed to authenticate and improving the data integrity. The performance of the proposed mechanism has been measured considering computational cost as a performance metric. The main drawback of the work is differential privacy still not included and vertical partition system has not been used to partition the data (Huang et al. 2017).

III. PROPOSED MECHANISM

The proposed framework has been divided into three frames to securely transmit and receive the data in a cloud environment.

1. Dice Coefficient and Cosine Similarity to compute the similarity
2. Use of encryption key to enhance the security
3. Data partitioning

Similarity Measures

A similarity measure which assigns a real value to sensitive data elements in a cloud, defining a resemblance between data files within range of 0 and 1. Similarity measure actually forms the base for many data partitioning techniques. Although, similarity measure compares the vector value which is symmetrical in nature and assigns a large value when these are similar.

Dice Coefficient

Dice Coefficient actually similar to Jaccard measure as both has been used to compute the similarity between the data elements. The basic formula to determine the similarity using Dice Coefficient is given as:

$$Dice\ Coefficient = \frac{2\lfloor X \cap Y \rfloor}{|X| + |Y|} \quad \epsilon \quad 0,1 \qquad (5)$$

In the given equation 5, X represents the data file and Y signifies the computed similar data file. Both files are of similar size with different data elements. The data elements in these files represented by $x_i and\ y_i$ respectively. In the given case, the given equation computed as follows:-

$$|X \cap Y| = \sum_i x_i y_i \qquad (6)$$

$$|X| = \sum_i x_i \qquad (7)$$

$$|Y| = \sum_i y_i \qquad (8)$$

The similarity between data elements represents the closeness and text based similarity does not recommended as it is precise and cannot be trusted. The proposed framework uses Dice Coefficient and Cosine similarity to determine the closeness between data elements. Algorithm 1 shows the mechanism of Dice Coefficient. The proposed algorithm uses both Dice Coefficient and Cosine Similarity as follows.

Algorithm 1 Dice Coefficient Relation {Function Create_Combined_Dice_Coeff (Improve_Sim)}

```
1. $For\ p\ = 0: Total\_Coincide\_Values$
2. $Dice\ Relation\ with\ data\ in\ cloud\ (DR) = Dice\_Relation(p);$
3. $DC = Dice\_Coefficient\_Relation(p)$
4. $Enhanced\_Similarity = DR/DC;$
5. $Combined\_Similarity(p, 1) = Main\_Data;$
6. $Combined\_Similarity(p, 2) = Connecting\_Data;$
7. $Combined\_Similarity(p, 3) = DC;$
8. $End\ For$
9. $Return\ DC;$
```

End Algorithm

Cosine Similarity

This type of similarity has been used when the angle between two elements and vectors matches in a meaningful way, normalized product of two vectors considered as appropriate similarity measure. Therefore, X and Y represented as vectors, the similarity of X and Y corresponds to relation between term vectors. This is represented as cosine angle between the vectors which is also called cosine similarity. It is the most common similarity measure used for various applications such as clustering (Eddamiri, 2019).

$$Cos\,\theta = X.Y\|X\|\|Y\| \qquad (9)$$

Algorithm: 2 Cosine Similarity Relation

```
1.  $Cosine\_data\_sim = function\ Cosine_{data}Similarity(repository)$
2.  $//$ $repository\ is\ the\ input\ whereas, Cosine\_data\_sim\ is\ the\ O/P$
3.  $Cosine\_data\_sim = [\,];$
4.  $Similarity\_count = 0;$
5.  $For\ z = 0\ to\ repository.length$
6.  $present_{data} = repository\ (z);$
7.  $For\ L = c + 1\ to\ repository.length$
8.  $S = |Cosine\_data(present_{data}) - cos(repository(z))|;$
9.  $Cosine\_data\_sim[relation\_count, 0] = present_{data};$
10. $Cosine\_data\_sim[relation\_count, 1] = repository(z);$
11. $Cosine\_data\_sim[relation\_count, 2] = S;$
12. $Similarity\_value = Similarity\_value + 1;$
13. $End\ for$
14. $End\ for$
15. $End\ function;$
```

End Algorithm

The similar values between data elements has been computed by following these algorithms and similarity measure has been calculated. The given figure 3 express the relation between the data elements such that X1 is directly linked with X2, X3, X4, and X5. Similarly, X2 at the sending side linked with X3, X4, and X5 at the receiver side. The similar pattern follows for the other data elements. Thus, similar data elements placed in one partition and other data elements has been placed in other partition. Thus, sensitive data similar in nature placed in one partition while general data placed in other blocks.

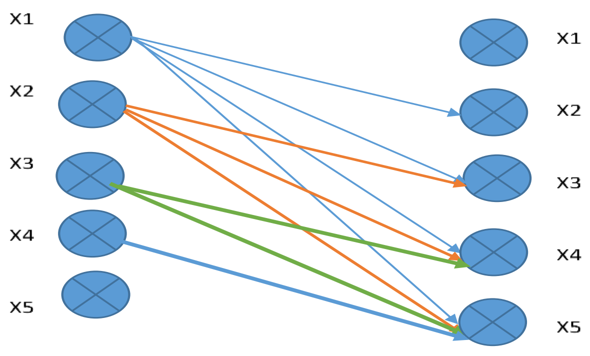

Fig. 3. Relation between data elements

The combined similarity measure consider dice coefficient in the numerator and cosine similarity in the denominator. However, storage structure for the files almost same.

Encryption Key

The primary key has been generated to enhance the security level. The data requested by the user from the cloud server has been encrypted before transmission. The encryption has been done by generating a unique key.

Algorithm 3: Data Encryption

```
1.  $Multiply\ Item\ Positions -$
    $Initially,\ an\ empty\ array\ has\ been\ generated\ to\ record\ values$
2.  $Initial\_Key = Dataset.InitialValue\ //$
    $the\ initial\ generated\ key\ consider\ as\ reference$
3.  $A\ (Counter) = 0;$
4.  $Dataset\ having\ columns\ and\ rows\ //$
    $Consider\ each\ column\ of\ the\ dataset$
5.  $If\ Dataset.Firm.Unique\ item\ value == Initial\_Key\ //$
    $When\ the\ column\ value\ is\ equal\ to\ the\ reference\ value$
6.  $A = A + 1;\ //\ The\ value\ has\ been\ incremented$
7.  $Item\_Position[A] = columnvalue;$
8.  $End\ If$
9.  $End\ For$
10. $Repeat\ till\ all\ the\ data\ has\ not\ been\ encrypted.$
```

End Algorithm

Data Partitioning

The encrypted value placed in data partition system in which combined relational value of dice coefficient and cosine similarity consider as an input for the data encryption system. The encrypted output further consider as input for the data partition system. The partitioning system process the encrypted data and place them in blocks to form the scalability. The given Algorithm 4 depicts the data partition.

Algorithm: 4 Data Partition Formation

```
$[Block1, Block\ 2, Block\ 3\ldots] = Initial\ partition\ formed\ in\ Blocks\ [Function\ created]$
$Block\ 1 = [\,];\ / / Initially\ partitions\ formed\ in\ Blocks\ which\ are\ empty$
$Block\ 2 = [\,];//\ Block\ 2\ also\ remain\ empty$
$Block\ 1\ Components$
        $= 0;\ //Total\ components\ in\ Block1$
$Block\ 2\ Components$
        $= 0;\ //Total\ components\ in\ Block2$
$Referred\ Component = Relationvalues(1,1)\ / / 1st\ component\ has\ been\ considered\ as\ referenced\ and\ then\ every\ element\ has\ been\ referred\ from\ this\ element.$
$Connection\ between\ Relationvalues\ / / The\ components\ having\ identical\ relation$
2.    $Identical\_Linkage = Compute(Relationvalues(:,1) == Referred\ Component);\ //$
3.    $Identical\_Linking\_Value = Sum(All\_Identical\_Linkage)/\ Total\ count\ of\ links//\ Compute\ the\ average\ value$
```

```
4.    $When\ Avg\ value\ of\ links > Identical\_links$  // $When\ value\ of\ other\ referred\ value\ greater\ than\ the\ component\ then\ it\ simply\ shifts\ to\ the\ referenced\ component\ block.$
5.    $Block1[Blockcomponents1] = Present\_File;$
6.    $Blockcomponents1 = Blockcomponents1 + 1;$
7.    $Else$
8.    $Block2\ [Blockcomponents2] = Present\_File;$
9.    $Blockcomponents2 = Blockcomponents2 + 1;$
10.   $End$
```

End Algorithm

The developed partition algorithm performs the operation to convert the large data files into small blocks. During the first time, the reference value has been set and then components of the block divided builds the formation of partition system.

Table I. Components corresponding value

Data in cloud	Linkage Data Value (LDV)	Corresponding Value
1	3	.24
1	2	.23
1	4	.63
1	3	.22
2	5	.32
2	3	.31

Table 1 represents the data in cloud which is connected to linkage data having LDV with a relation. In the given table, it is clear that data value 1 is connected to data value 3 with its corresponding component value is .14. Similarly, data value 1 is linked to data value 2 and data value 4 so the corresponding reference value is (.23 + 0.64)/2=0.43 which is greater than reference corresponding value 1. Hence, components of data value 1 and data value 2 does not placed in the Cluster 1 and Cluster 2. Thus, now this value 0.43 considered as reference value for the system. Therefore, once partitions has been created then metaheuristic algorithm has been applied to optimize the data. The formation of metaheuristic algorithm based on the swarm behavior of the bees (onlooker, employed and scout). The algorithm named as Artificial Bee Colony algorithm used by many researchers to solve the complex problems. In this paper, ABC optimize the data placed in the partition system.

Algorithm 5: ABC Algorithm

```
1   Total Data in partitions has been initialized
2   Initialize the bee categories-
        a. Employed bee
        b. Onlookers bee and
        c. Scouts bee
3   Threshold value has been set = ft
4   The scout bee value set as fs
5   For M=1 → partitions in cloud
6   The objective value R has been called
7   $ABC_{obj} = \begin{Bmatrix} M_{Bee} > O_{Bee}\ then\ R\ True \\ M_{Bee} \leq O_{Bee}\ then\ R\ false \end{Bmatrix}$
8   Return $Z_{Bee}R$
9   Data has been optimized = $ABC\ (ABC_{obj}, f_s, f_t)$
10  End
11  $Z_{Bee} = Data\ to\ be\ optimized$
12  Return optimized data and prevent the partitions in cloud
```

End Algorithm

ABC algorithm take partition data as input and compute the unique value in the list and each unique component has a certain weight. The level of the data has been set accordingly based on the optimized data occurrence. The proposed method has been cross validated using the Artificial Neural Network. ANN is widely used by many researchers and practitioners to solve the various problems. It is a three layer structure namely input layer, hidden layer, and output layer. The input layer consider raw data as input and corresponding weight has been generated by a hidden layer. The data has been travelled in iterations and weight has been changed accordingly. Consequently, ANN trained the weighted data which is further cross validate the input data.

Algorithm 6: ANN Algorithm for cross validation

```
Input: $Network\ properties\ as\ a\ Training\ Data\ (T_D), Target\ (T_g)$
Output: $Best\ Data\ Partition\ to\ protect\ the\ sensitive\ data$

Initialize ANN with parameters
– $Epochs\ (E)$
– $Neurons\ (N_s)$
– $Performance\ parameters: Mutation\ and\ Validation$
– $Training\ Techniques: Levenberg\ Marquardt\ (Trainlm)$
– $Data\ Division: Random$
For Training data
    $G = Training\ data\ categorised$
End
$ANN\ has\ been\ initialized\ using\ T_D\ and\ G$
$Value = Newff\ (T_D, T_g, N_s)$
$Set\ the\ training\ parameters\ according\ to\ the\ requirements$
$Net = Train\ (Value, T_D, G)$
$Classify = simulate\ (Value, blocks\ component)$
$If\ Classify = True$
$Best\ partition = Simulated\ data$
Else
$Data\ is\ not\ partitioned = Simulated\ data$
```

The simulated training approach tests and train the raw data taken as input from data partition system. The trained and test data if changes more than 25% that of trained structure then data has been accepted in a correct blocks and partitions else partition in the cloud further readjusted.

IV. Results

The results has been computed considering four parameters listed as follows:-

$$True\ Positive\ (T_P) = \frac{Total\ true\ selected\ components}{Total\ size\ of\ samples} \quad (10)$$

$$False\ Positive\ (F_P) = \frac{Total\ False\ selected\ components}{Total\ size\ of\ samples} \quad (11)$$

$$True\ Negative\ (TN) = \frac{Total\ remaining\ true\ samples}{Total\ size\ of\ samples} \quad (12)$$

$$False\ Negative\ (FN) = \frac{Total\ remaining\ false\ samples}{Total\ size\ of\ samples} \quad (13)$$

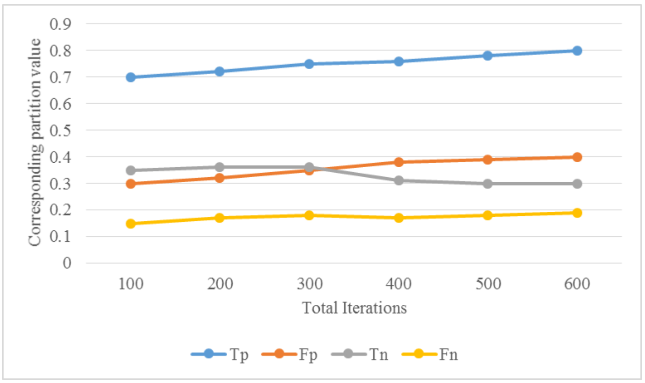

Fig 4. Evaluated results computed considering the true and false rates

Fig. 4 clearly shows that the high true positive value represents the good partition of the data. The high TP signifies the data has been placed in a good partition cluster and there is not any problem in retrieving and modifying the data. The proposed framework has been passed 600 iterations to partition the data. The average value obtained for the TP is 0.751. However, the average value obtained for the false positive is 0.35. False positive signifies the bad selection of the partition value. If the value of false positive is large then it represents that data placement was not good in an appropriate blocks. The obtained average value of 0.35 for the false positive signifies that there is not any false placement of the data. Consequently, Tn represents the total bad placement data samples which is a controversial factor. The large value of Tn shows that large number of bad data samples is good for different number of iterations but on the same side, it also shows that irrelevant data placed in blocks. The obtained true negative value is near to false positive which is 0.31. Similarly, false negative of the proposed structure is 0.17 which represents the negative selection of the data in the partition system. The low partition value represents that negative placement of data is very less. Thus, proposed framework holds good overall partition of the data in a cloud.

V. CONCLUSION

In this paper, an efficient cloud security data partitioning framework has been proposed to provide scalability and consistency. The data partitioning enables the system to store the data in an easy manner. The data storage possible using these techniques in an effective manner within less cost. In addition, the proposed structure effectively reduces the storage space by efficiently partitioning the data. The similarity measures determine the similarity between the data elements. The sensitive data which is identical placed in one partition and other data placed in second partition and so on. In this way, an effective and reliable partitioning system has been developed which reduces the load and provides scalable data. The proposed structure has been validated using the metaheuristic approach ABC to optimize the partitioned data. The results has been further cross validated using the ANN to determine its effectiveness. The experiments results measured in terms of true and false negative and positive value. The obtained true positive value is 0.75 which shows that data has been effectively partitioned in blocks and false negative value 0.17 depicts that very less bad partitioning has done. Overall, proposed work effectively partitioned the data in a cloud environment.

REFERENCES

[1] Ahirrao, S. and Ingle, R., 2015. Scalable transactions in cloud data stores. Journal of Cloud Computing, 4(1), p.21.
[2] Khedkar, S.V. and Gawande, A.D., 2014. Data partitioning technique to improve cloud data storage security. International Journal of Computer Science and Information Technologies, 5(3), pp.3347-3350.
[3] Li, J., Huang, L., Zhou, Y., He, S. and Ming, Z., 2017. Computation partitioning for mobile cloud computing in a big data environment. IEEE Transactions on Industrial Informatics, 13(4), pp.2009-2018.
[4] Zhang, Z., Deshmukh, H. and Patel, J.M., 2019. Data Partitioning for In-Memory Systems: Myths, Challenges, and Opportunities. In CIDR.
[5] Levitin, G., Xing, L. and Dai, Y., 2017. Optimal data partitioning in cloud computing system with random server assignment. Future Generation Computer Systems, 70, pp.17-25.
[6] Xing, L. and Levitin, G., 2017. Balancing theft and corruption threats by data partition in cloud system with independent server protection. Reliability Engineering & System Safety, 167, pp.248-254.
[7] Huang, C., Lu, R. and Choo, K.K.R., 2017. Secure and flexible cloud-assisted association rule mining over horizontally partitioned databases. Journal of Computer and System Sciences, 89, pp.51-63.
[8] Shinde, Y. and Vishwa, A., 2015. Privacy preserving using data partitioning technique for secure cloud storage. International Journal of Computer Applications, 116(16).
[9] Kanade, A., Mule, R., Shuaib, M. and Nagvekar, N., 2015. Improving cloud security using data partitioning and encryption technique. International Journal of Engineering Research and General Science, 3(1).
[10] Shaikh, M., Achary, A., Menon, S. and Konar, N., 2015. Improving cloud data storage using data partitioning and data recovery using seed block algorithm. International Journal of Latest Technology in Engineering, Management & Applied Science, 4(1).
[11] Eddamiri, S. and Benghabrit, A., 2019. An improved RDF data Clustering Algorithm. Procedia computer science, 148, pp.208-217.
[12] Nadaph, A. and Maral, V., 2014. Cloud computing-partitioning algorithm and load balancing algorithm. International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), 4(5).