Section A-Research paper
ISSN 2063-5346

Enhanced Monarch Butterfly Optimization based MapReduce Safely using Edwards Curve(MBO-MR-Ed25519) Algorithm for Securing data

S.Syed Nawas Husain[1] and Dr. R.Balasubramanian[2]

[1]Research Scholar, [2]Professor
[1]Reg No: 20114012281034
Department of Computer Science and Engineering
ManonmaniamSundaranar University, Abishekpatti, Tirunelveli-627012

ABSTRACT
The security of data is enhanced by using techniques such as encryption and compression to enhance security. Health care, and more specifically cardiovascular disease, has been extensively researched with cryptographic methods. This security is focused on cardiac disease in order to focus on the methods as much as possible. Various optimization problems can be solved with Monarch Butterfly Optimization (MBO) based MapReduce (MR) is a depiction of encoding expected to produce vast datasets. To enhance the security of Ed25519 algorithm, a new transmission security processing mechanism which consists of three phases - key generation, signature generation, and signature verification - is proposed in this paper. In this paper we propose a model of (MBO-MR-Ed25519) for the optimization process using the Monarch Butterfly Optimization algorithm based MapReduce for parallel execution. Finally, the Ed25519 algorithm has been applied to secure the Cardiovascular data. The experimental results show that our proposed method produces relatively stable parallel results, has a faster operating speed, less time and effectively high security.

Keywords: MBO, MapReduce, Ed25519, Key Generation, Signature Generation and Verification.

1. INTRODUCTION

There has been a dramatic increase in the volume of data transmitted over the internet, according to a recent study on data transmission security. New technologies, devices, and communication channels, like social networking sites and scientific research, have increased the volume of digital information. Due to their importance to the day-to-day functioning of patients, Cardiovascular Records are very confidential data [14]. Newly generated individuals are passed on to the next generation regardless of whether they have a higher fitness level or not [4]. In Map-Reduce, more files are stored and a read-write operation is performed to accomplish the task [20]. Ed25519 uses a key pair with a private key and public key to verify the correct and incorrect signatures for each mod size selected [24]. An equation based on cryptography with keys is very efficient [22].

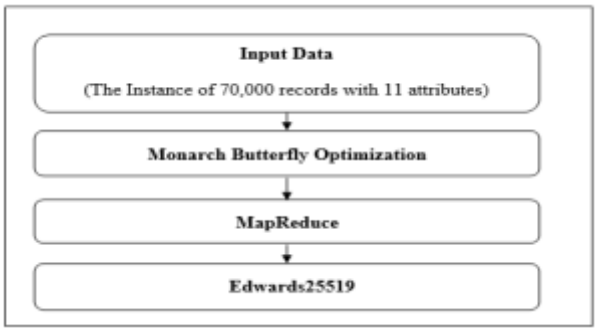

Figure 1: Overall Architecture

1.1 Monarch Butterfly Optimization

The Monarch Butterfly Optimization (MBO) [3], the process is simple and easy to implement. Subpopulation1 and Subpopulation2 are two equal-sized subpopulations in MBO. In subpopulation 1, the half of the individuals with the best fitness value constitutes subpopulation 1, and the other half constitutes subpopulation 2.As a result, MBO has two strategies: migration operators and butterfly operators.

1.1.1 Migration

By simulating monarch butterfly migration, solutions can explore different areas and potentially find better solutions. By exchanging information between different regions, the migration operator maintains population diversity. This facilitates exploration of new areas and prevents premature convergence by enabling solutions to move into unexplored areas [2].

1.1.2 Adjusting

Using it adjusting operator, the solutions within a particular region are adjusted or modified [14]. It refines and improves the quality of individual solutions within a specific region by working on individual solutions. In addition to promoting exploitation, it enhances the quality of the solutions within a region [2].

1.2 MapReduce

An array of MapReduce programs and computational frameworks can be used to handle large data volumes in a distributed and parallel manner. An input value pair that is mapped into an intermediate value pair is known as a mapper. By dividing these yields by reducer, mappers are organized. Using MapReduce, fault tolerance can be achieved [20].

1.2.1 Types of Format in MapReduce

A MapReduce Input Format and Output Format in Hadoop can be used for a variety of purposes [19].

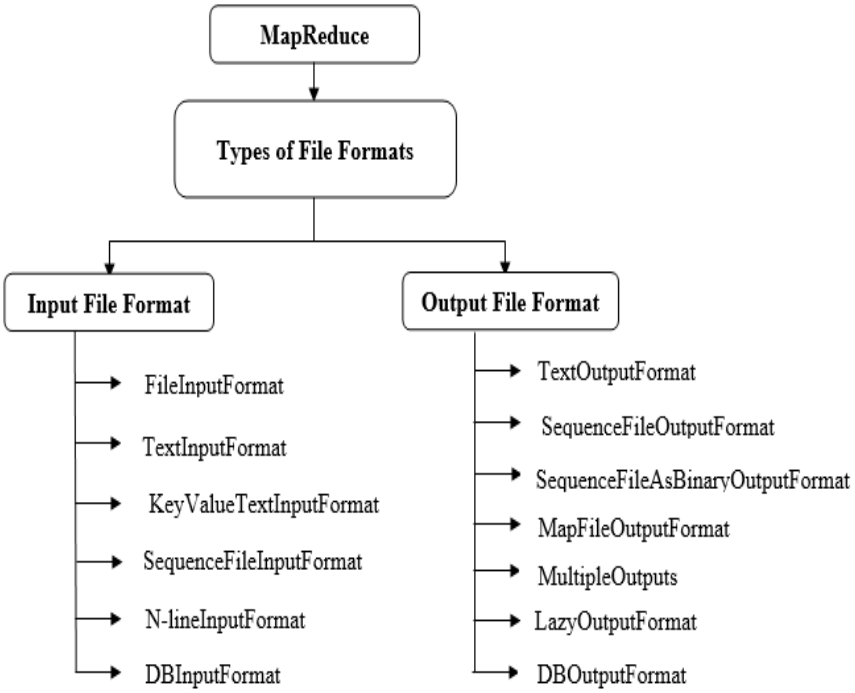

Figure 2: Types of Format

1.2.2 Input File Format:

Input Format explains how Map-Reduce jobs are input-specified for execution. The InputFormat defines how these input files are to be split and read. Input files or other objects are selected [19].

Table I: Input File Format in MapReduce

Input Format	Description
FileInputFormat	Base class for all file-based InputFormats. Reads a path containing files and divides them into InputSplit
TextInputFormat	Default InputFormat. Treats each line of input file as a separate record without parsing. Suitable for unformatted d or line-based records like log files.
KeyValueTextInputFormat	Treats each line of input as a separate record and breaks it into key-value pairs.
SequenceFileInputFormat	Reads sequence files, which are block-compressed binary files storing sequences of key-value pairs with direct serialization/deserialization.
NLineInputFormat	Variant of TextInputFormat where each mapper receives a fixed number of lines as input.

DBInputFormat	Reads data from a relational database using JDBC, and suppo loading small datasets for joining with large datasets.

1.2.3 Split

Each Mapper will process a set of data. A map task is created for each split. Hence, the number of map tasks is equal to the number of input splits. Split is divided into records, which are processed by the mapper [20].

1.2.4 Read Record

A communication is established between the inputSplit and the outputSplit. The data is then converted into key-value pairs that can be read by the Mapper. The RecordReader converts data into a key-value pair by default using TextInputFormat. InputSplit communicates with it until the reading of the file is complete. Each line in the file is assigned a byte offset. A mapper then processes the key-value pairs [16].

1.2.5 Mapper

Input records are processed by RecordReader and intermediate key-value pairs are generated. Intermediate outputs are completely different from inputs. As a result of the mapper, a collection of key-value pairs is produced [15].

1.2.6 Shuffling and Sorting

The output of [17] is shuffled to the reduce node after partitioning. The shuffling of data occurs over the network. Once all the mappers have finished and shuffled their output, the reducers can begin. This intermediate output is then merged and sorted by the framework. In the reduce phase, this is used as input.

1.2.7 Reducer

An intermediate key-value pair is then provided to the reducer by the mappers as an input. The output is then generated by running each reducer function. Final output [16] comes from the reducer.

1.2.8 Record Writer

In the Reducer phase, this output key-value pair is written to the output files [19].

1.2.9 Output File Format:

Map-Reduce jobs are executed according to the Output Format specification. These key-value pairs are written in output files by RecordReader using the OutputFormat [18].

Table II: Output File Format in MapReduce

Output Format	Description
TextOutputFormat	Default OutputFormat for MapReduce jobs. Writes (key, value) pairs on individual lines of text files.
SequenceFileOutputFormat	Writes output as sequence files. Serializes arbitrary data types and preserves their serialization for the next mapper.
SequenceFileAsBinaryOutputForma	Variant of SequenceFileOutputFormat that write keys a values to sequence files in binary format.

MapFileOutputFormat	Writes output as map files, with keys added in order. Requires reducer to emit keys in sorted order.
MultipleOutputs	Allows writing data to files with names derived from output keys and values.
LazyOutputFormat	Wrapper OutputFormat that prevents the creation of empty output files in some cases.
DBOutputFormat	OutputFormat for writing to relational databases and HBaseand accepts key-value pairs, where the key extends DBWritable.

1.3.1 Edwards25519

Ed25519, you combine EdDSA and Curve25519 into one elliptic curve signing algorithm [21]. ECDLP is a computational challenge based on the Schnorr signature algorithm. It is considered the fastest algorithm across various metrics due to its implementation on the 255-bit Curve25519 [22]. Public key encryption is used to compute and authenticate digital signatures, such as Ed25519 [23]. Documents are attached to it for verification after they are transmitted electronically. In EdDSA, Bernstein et al. developed Edwards curves, especially edwards25519 (Ed25519), which is a twisted Edwards curve. It is necessary to instantiate specific parameters for EdDSA to function properly [25].

2. LITERATURE REVIEW

2.1 Monarch Butterfly Optimization

The MBO algorithm simulates monarch butterfly migration by exchanging information between monarch butterflies. The MBO consists of simplification and idealization before the butterfly individuals are grouped into Land1 and Land2, and the butterfly's position is updated by migration and adjustment afterwards. Concurrent adjustment and migration operations allow the MBO to be processed in parallel while taking into account global and local search capabilities [4]. The literature underscores the pivotal role of security in digital transformation, particularly in data encryption algorithms with a focus on key length and settings. Hybrid (MBO-Ed25519) Monarch Butterfly Optimization (MBO) emerges as a robust tool for intricate optimization problems, and its fusion with Ed25519 showcases promise in expediting factorization through optimized calculations. The proposed approach exhibits practical significance for both optimization and cryptographic domains, enhancing computation efficiency and meeting the multifaceted expectations of secure data exchange [25].MBO is divided into two equal subpopulations, subpopulation 1 and subpopulation 2. The individuals in subpopulation1 update their positions through migration operators, and those in subpopulation2 adjust their positions through butterfly adjusting operators[3]. Due to the performance of [2] BOA, several optimization problems have been solved using it.

2.2 MapReduce

The purpose of this paper is to classify a dataset containing student's data using mapeduce algorithms in big data tools. A complex and difficult task is to manage and analyze numerous datasets using HDFS and MapReduce along with evaluating performance [6]. For processing learning management system log file data, this paper invoked MapReduce and Hadoop applications. Through

trustworthiness models, students' activity data in collaboration and security in e-Learning can be analyzed using the LMS log data [8]. UOC courses use a MapReduce approach based on collaborative learning activities, and the amount of data is growing rapidly. Using mapreduce, we developed a method for on-line e-assessment. As a new proposed Secured MapReduce (SMR) layer, this paper introduces a security and privacy based layer between HDFS and MR Layer (MapReduce), and this model is called SMR. Compared to existing approaches, this SMR model has remarkable improvements in running time and information loss, as well as optimized CPU and memory usage [7].

2.3 Hybrid MapReduce

AES-MR is the new combination of AES and MapReduce for securing HDFS data. The AES-MR encryption algorithm only combines the AES encryption algorithm and the MapReduce parallel programming paradigm [7]. By using the MapReduce programming model, it is easier to incorporate the AES encryption algorithm to work in parallel and save a lot of time. As part of our work, MapReduce will be used to encode large data volumes and vital data by using the AES encryption algorithm as part of XTX mode [9]. A novel blowfish-based algorithm is used to encrypt and decrypt data in this paper. By encrypting and decrypting data using the Hadoop cluster and MapReduce, the cost of encryption has been reduced significantly. Considering security, time consumption, and performance, the proposed algorithm achieves a balance. Observing that the proposed method for encrypting the data increases the power and safe consumption by about 50%, where the percentage increases with increase in encrypted file size. Because of that, it had the best performance [1].

2.4 Edwards25519

A study of the optimization and implementation of elliptic curve cryptographic algorithms is presented in this paper. We implement cryptography-specific and SIMD extensions to our instruction set architecture. Based on AVX2 vector instructions, the software implementations demonstrate superior performance for X25519, X448, Ed25519, and Ed448 protocols [5]. Furthermore, trade-offs and limitations were identified, providing valuable insights for future advancements. In this work, we aim to inspire students and researchers to conduct future research in this area [10]. Data integrity and security are crucial, which is why EdDSA is adopted for digital signatures and verification, and SHA-256 is adopted for cryptographic hashes [11]. In order to evaluate the efficiency of a system, performance metrics such as transaction speed, per transaction time, and the time required to sign and verify transactions are commonly used [12]. The review compares different systems in terms of privacy, transaction costs, large file storage, implementation, and registration costs [13].

3. METHODOLOGY

Hospitality orientation can be a very challenging task, and it takes a lot of time for Cardiovascular Records and their families, because it depends on several factors, including diastolic blood pressure, cholesterol, etc. For a good result, we need to collect a large amount of data and have a significant impact on the patient's records path, and we need a powerful data security tool in order to secure the data. Optimizing problems have become increasingly complex in recent years, which promotes research into improving optimization algorithms. Monarch Butterfly Optimization, however, used this method to solve problems and get an optimal value; so, we used the MapReduce framework using a distributed and parallel programming model to compress large amounts of data. An another hand, Cardiovascular Records data through Map reduce and retrieve the data safely using Ed25519 algorithm.

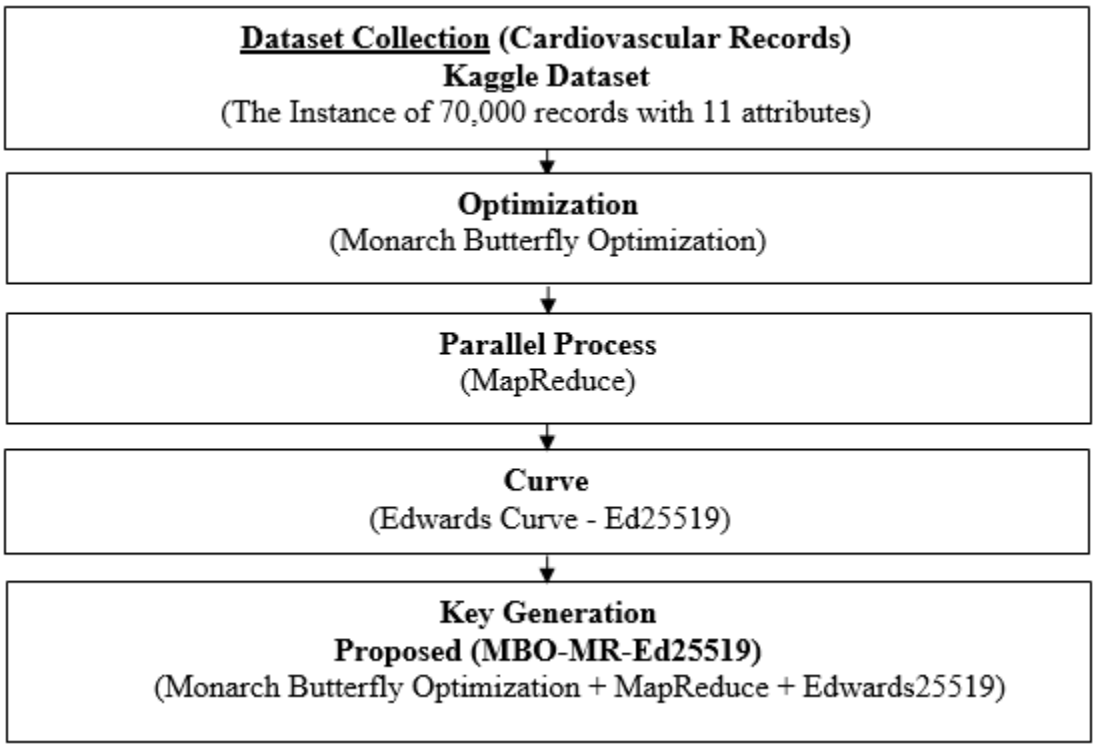

Figure 3: Proposed Architecture

3.1 Monarch Butterfly Optimization (MBO)

It represents a unique perspective and approach to optimization, the monarch butterfly optimization algorithm (MBO). In an optimization problem, it combines exploration, exploitation, and return phases to find optimal solutions using nature-inspired optimization techniques. A Monarch Butterfly Optimization algorithm balances exploration and exploitation while maintaining population diversity to find optimal or near-optimal solutions. In monarch butterfly optimization, there are two types of operations used to maximize the statistical value (dataset). Our focus here is on adjusting and migrating operators.

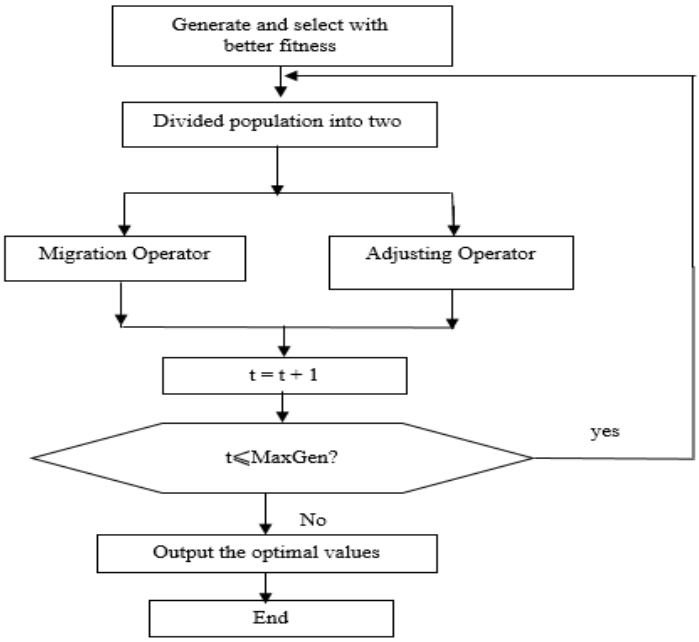

Figure 4: Monarch Butterfly Optimization

Algorithm 1: Monarch Butterfly Optimization

Step 1: Set the population *NP*, max generation *MaxGen*, dimention *D*, the Max size S_{max}, adjusting rate *BAR*, migration *peri* and migration rate *p*. let counter t=1.
//Initialization operation
Step 2: Generate opposition population according to OBL. Select the individuals with better fitness to enter the next generation from the original and opposition based populations.
Step 3: Calculate fitness values according to location of monarch butterfly
//Fitness evaluation
Step 4: while
t ≤ *MaxGen do*
Sort the population according to monarch butterfly fitness using quicksort algorithm.
Divide the monarch butterfly population into two subpopulation
(i)Subpopulation 1 and (ii) Subpopulation 2
for *j=1* to *Np1* do
update subpopulation 1 using migration operator
end *for*
for *j=1* to *Np2* do
update subpopulation 2 using adjusting operator
end for
Merge two new subpopulation into a new population
Recalculate the fitness values of each monarch butterfly according updated position
Let t = t + 1.
Step 5: end while
Step 6: output the optimal values

3.2 MapReduce

MapReduce is an algorithm and a method for distributed computing based on the Java programming language. Map and Reduce are two important tasks in the MapReduce algorithm. Data is converted into tuples (key/value pairs) by mapping a set of data into another set of data. Second, we have the reduce task, which takes the output from a map and uses it to combine those data tuples. Reducing is always carried out after the map step, as indicated in the name MapReduce.

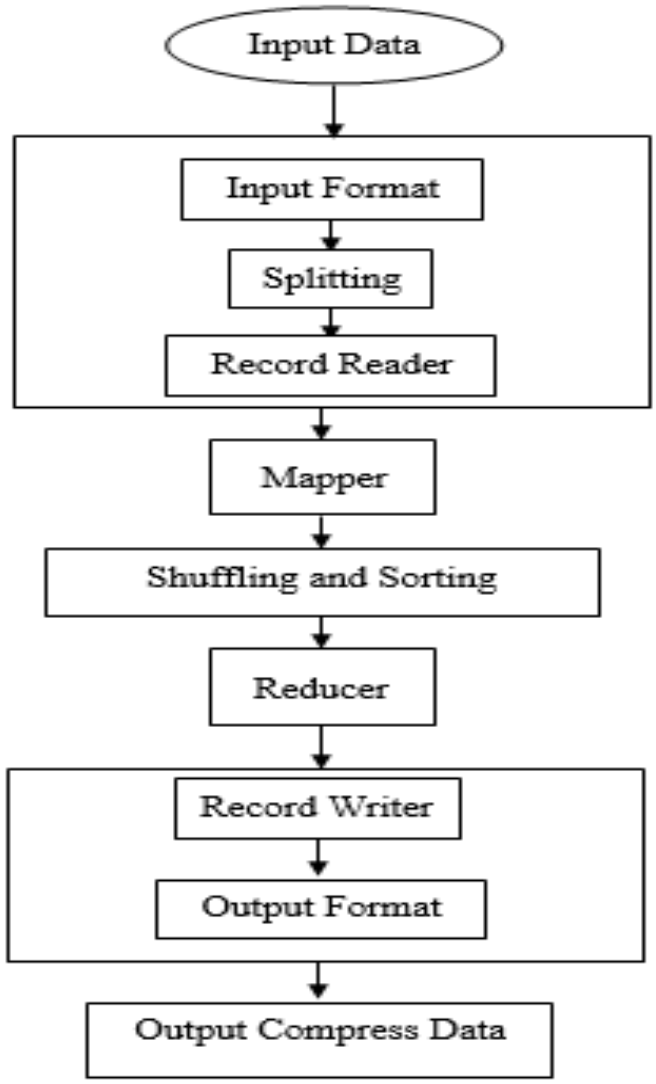

Figure 5: MapReduce Architecture

- **Map stage** − During the map stage, the mapper or map is responsible for processing the input data. Mapper functions receive input files line by line. Several small chunks of data are created as the mapper processes the data.
- **Reduce stage** − In this stage, the Shuffle and Reduce stages are combined. Mapper data is processed by the Reducer.

Algorithm 2: MapReduce

```
Step1: Map
        map_outputs = [ ]
      for data_item in dataset:
            // Apply mapping function to each data item
        mapped_data = MapFunction(data_item)
            // Add mapped data to map_outputs
        map_outputs.append(mapped_data)
    return map_outputs
Step 2: ShuffleAndSort (map_outputs):
            // Group map outputs by key
     key_value_pairs = GroupByKey(map_outputs)
            // Sort key-value pairs
     sorted_key_value_pairs = SortByKey(key_value_pairs)
   return sorted_key_value_pairs

Step 3: Reduce (sorted_key_value_pairs):
     result = []
  for key, values in sorted_key_value_pairs:
            // Apply reduce function to values with the same key
    reduced_value = ReduceFunction(key, values)
          // Add reduced value to result
    result.append(reduced_value)
  return result
```

3.3 Edwards25519

The Ed25519 algorithm uses EdDSA and Curve25519 to sign elliptic curves. EdDSA relies on the difficulty of the ECDLP problem to base its signature algorithm on Schnorr's. 255-bit curves, such as Curve25519. Fast single-signature verification is one of the attractive features of Ed25519, a public-key signature system. All metrics indicate that Ed25519 performs the best.EdDSA is the most secure algorithm based on key length. A private/public key pair must be generated by the signer, with the private key for signing and the public key for verifying. In order to compute Ed25519 signatures, we combine the proof options hash and the credentials without proof hash. We use the private key to compute the combination hash.

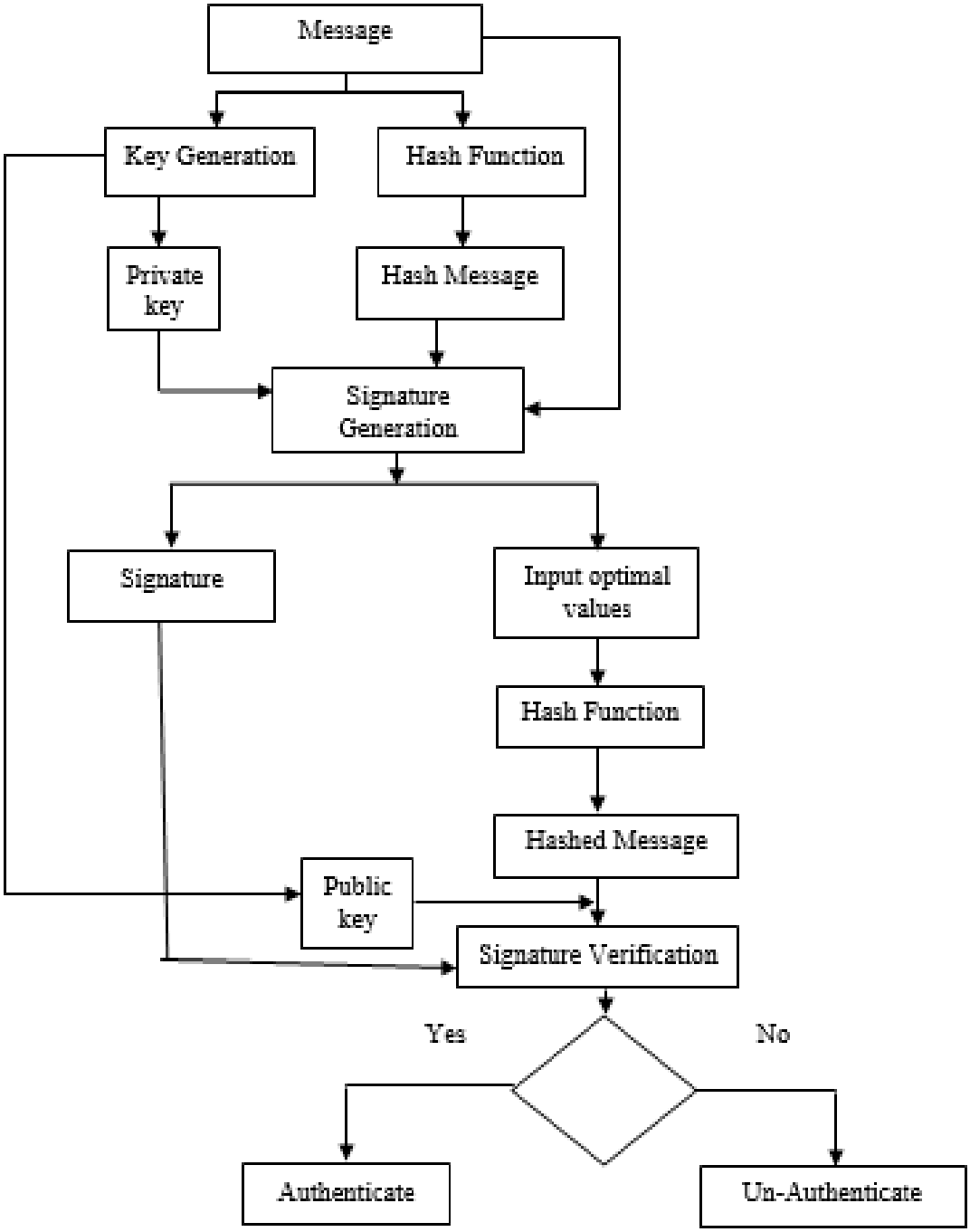

Figure 6: Ed25519 Diagram

Algorithm 3: Edwards25519

Step 1: Key Generation

- private key (integer): privKey
- public key (EC point): pubKey = privKey * G

Step 2: Signature Generation

- Calculate *pubKey* = *privKey* * G
- Deterministically generate a secret integer $r = \text{hash}(\text{hash}(privKey) + msg) \bmod q$
- Calculate the public key point behind r by multiplying it by the curve generator:

$$R = r * G$$

- Calculate $h = \text{hash}(R + pubKey + msg) \bmod q$
- Calculate $s = (r + h * privKey) \bmod q$
- Return the signature $\{ R, s \}$

Step 3: Signature Verification

- Calculate h = hash (R + pubKey + msg) mod q
- Calculate P1 = s * G
- Calculate P2 = R + h * pubKey
- Return P1 == P2
- Finally verify the message is Authenticated or Unauthenticated

3.4 Proposed (MBO-MR-Ed25519)

This work proposes a new model that combines Monarch Butterfly Optimization (MBO) with MapReduce (MR) safely using Edwards25519 (Ed25519), fully intent on signature verification before optimization. From the input data, the optimal value is compressed using MapReduce to obtain the output compressed data. The data is then secured using Ed25519 and optimized. In this paper, we propose (MBO-MR-Ed25519) as a technique that overcomes the disadvantages of each. The monarch butterfly optimization method is used to obtain optimal values for cardiovascular records. Creating mappers and reducers for data processing can be challenging. Data is finally compressed safely using Ed25519 algorithm.

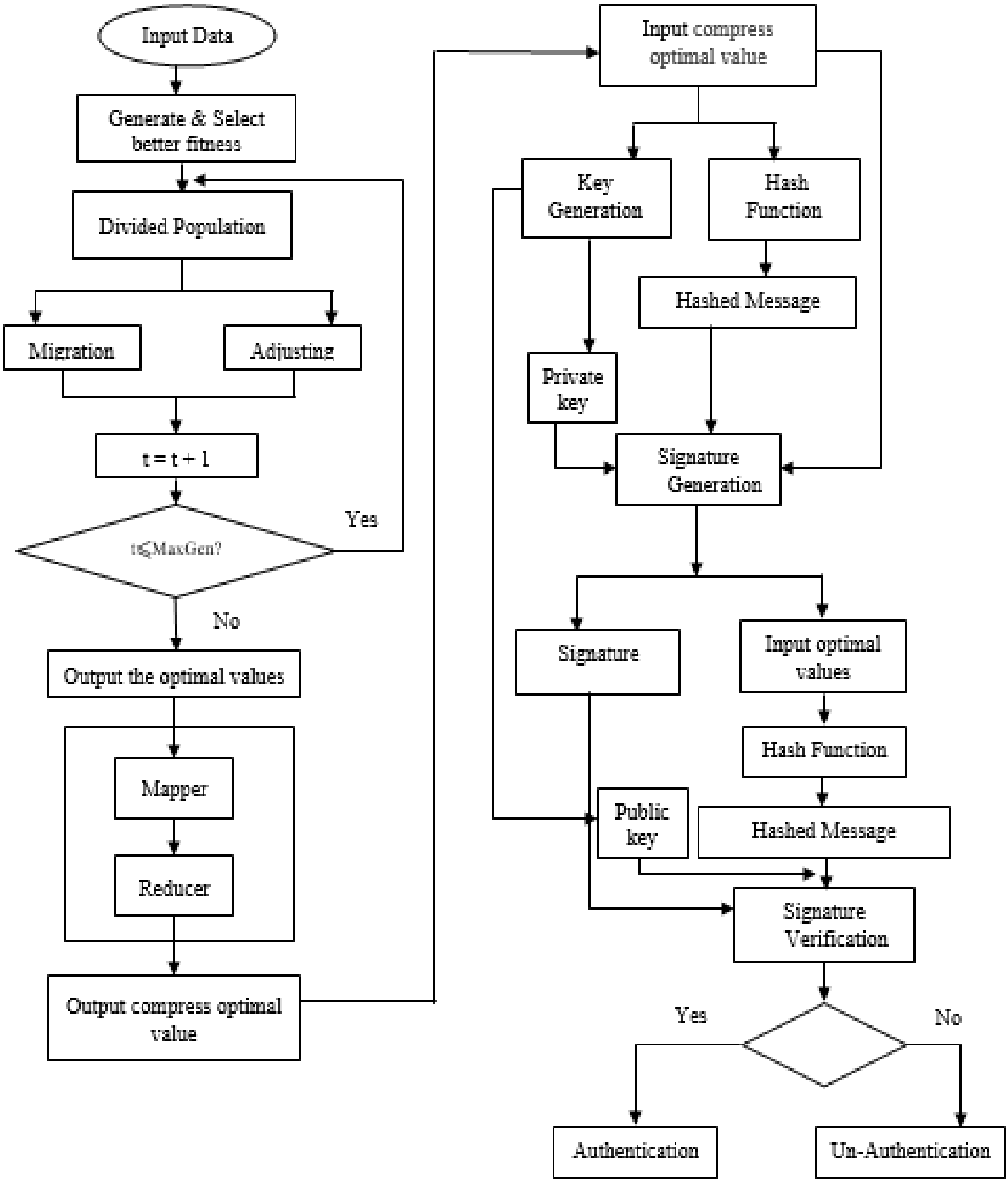

Figure 7: Proposed Flow Diagram of MBO-MR-Ed25519

Algorithm 4: Proposed (MBO-MR-Ed25519)

Input Data: Cardiovascular Records (Records – 70,000) with 11 Attributes
Step 1: Input data applied into monarch butterfly optimization to generate an optimal value.
Step 2: Initialization:
Set the generation counter $t = 1$; initialize the population P of NP monarch butterfly individuals randomly; set the maximum generation MaxGen, monarch butterfly number $NP1$ in Land 1 and monarch butterfly number $NP2$ in Land 2, max step SMax, butterfly adjusting rate BAR, migration period peri, and the migration ratio p.
Step 3: Fitness evaluation: Evaluate each monarch butterfly according to its position.
Step 4: While the best solution is not found or $t < MaxGen$ do Sort all the monarch butterfly individuals
according to their fitness. Divide monarch butterfly individuals into two subpopulations (Land 1 and Land 2);

- Generate new Subpopulation 1 according to migration operator end *for i for j= 1* to *NP2* (for all monarch butterflies in Subpopulation.
- Generate new Subpopulation 2 according to adjusting operator. end for j Hybrid the two newly-generated subpopulations into one whole population; Evaluate the population according to the newly updated positions; $t = t+1$.

Step 5: end while
Step 6: Output the optimal values.
Step 7: Get the optimal values applied into MapReduce, to compress a new data.

Step 8: Map (optimal values)

```
map_outputs = [ ]
for data_item in dataset:
        // Apply mapping function to each data item
    mapped_data = MapFunction(data_item)
        // Add mapped data to map_outputs
    map_outputs.append(mapped_data)
return map_outputs
```

Step 9: ShuffleAndSort (map_outputs):

```
    // Group map outputs by key
key_value_pairs = GroupByKey(map_outputs)
    // Sort key-value pairs
sorted_key_value_pairs = SortByKey(key_value_pairs)
return sorted_key_value_pairs
```

Step 10: Reduce (sorted_key_value_pairs):

```
result = []
for key, values in sorted_key_value_pairs:
        // Apply reduce function to values with the same key
    reduced_value = ReduceFunction(key, values)
        // Add reduced value to result
    result.append(reduced_value)
return result
```

Step 11: Output the compressed the result.
Step 12: Get the compressed the result applied into Ed25519, to generate a pair of keys.
Step 13: Key Generation

- private key (integer): *privKey*
- public key (EC point): *pubKey = privKey * G*

Step 14: Signature Generation

- Calculate *pubKey = privKey* * G
- Deterministically generate a secret *integer* ***r*** *= hash(hash(privKey) + msg) mod q*
- Calculate the public key point behind ***r*** by multiplying it by the curve generator:

$$R = r * G$$

- Calculate $h = \text{hash}(R + pubKey + msg) \bmod q$
- Calculate $s = (r + h * privKey) \bmod q$
- Return the signature $\{ R, s \}$

Step 15: Signature Verification

- Calculate *h = hash (R + pubKey + msg) mod q*
- Calculate *P1 = s * G*
- Calculate *P2 = R + h * pubKey*
- Return *P1 == P2*

Step 16: Finally verify the message is Authenticated or Unauthenticated

4. EXPERIMENTAL SETUP

4.1 Data Collection

The dataset used is obtained from the Kaggle repository online [14]. The dataset consists of 70,000 clinical trial records of patients' data collected by hospitals for cardiovascular related diseases, and there are three input features within the dataset: Objective (realistic-information), Examination (outcomes of medical investigation), and Subjective (data obtained from a patient). Also, the dataset

has11 attributes from which 4 are objective features, 4 examination features, 3 subjective features.The following Table III represents the types of attributes.

Table: III Types of Attributes [14]

S.No	Attributes	Input Features	Data Type/ Description
1	Age	Objective Features	Int / days
2	Height		Int / centimeters
3	Weight		Float / kilograms
4	Gender		Categorical code 1: male, 2: female.
5	Systolic blood pressure	Examination Features	Int/
6	Diastolic blood pressure		Int/
7	Cholesterol		1: normal, 2: above normal, 3: well above normal
8	Glucose		1: normal, 2: above normal, 3: well above normal
9	Smoking	Subjective Features	Binary
10	Alcohol		Binary
11	Physical activity		Binary

4.2 Performance Metrics

4.2.1 Encryption time:

$$Encryption\ time = \frac{Size\ of\ data\ to\ be\ encrypted}{Encryption\ speed} \quad (1)$$

4.2.2 Decryption time:

$$Decryption\ time = \frac{Size\ of\ data\ to\ be\ decrypted}{Decryption\ speed} \quad (2)$$

4.2.3 Total time:

$$Total\ time = Encryption\ time + Decryption\ time \quad (3)$$

4.2.4 Throughput:

$$Throughput\ of\ encryption = \frac{Total\ plain\ text\ (bytes)}{Encryption\ time} \qquad (4)$$

4.2.5 Energy consumption:

$$Energy\ consumption\ (in\ joules) = Power\ (in\ watts)\ x\ Time\ (in\ minutes) \qquad (5)$$

4.3 Results

Encryption time, Decryption time, Total time, Throughput and Energy consumption is used to evaluate performance. Performance comparisons are performed on Ed25519 and MBO-Ed25519 when exiting. The proposed (MBO-MR-Ed25519) approach achieves less complexity execution of Total Time of 9.35(m).

Table IV. Performance Analysis

Algorithm	Number of Records	Encryption Time (m)	Decryption Time (m)	Total Time(m)	Throughput	Energy Consumption
Ed25519	70,000	8.7	7.95	16.65	51.9098	94.7754
(MBO - Ed25519)		5.7	5.85	11.55	67.4829	89.6857
Proposed (MBO-MR-Ed25519)		4.45	4.9	9.35	77.2795	79.9876

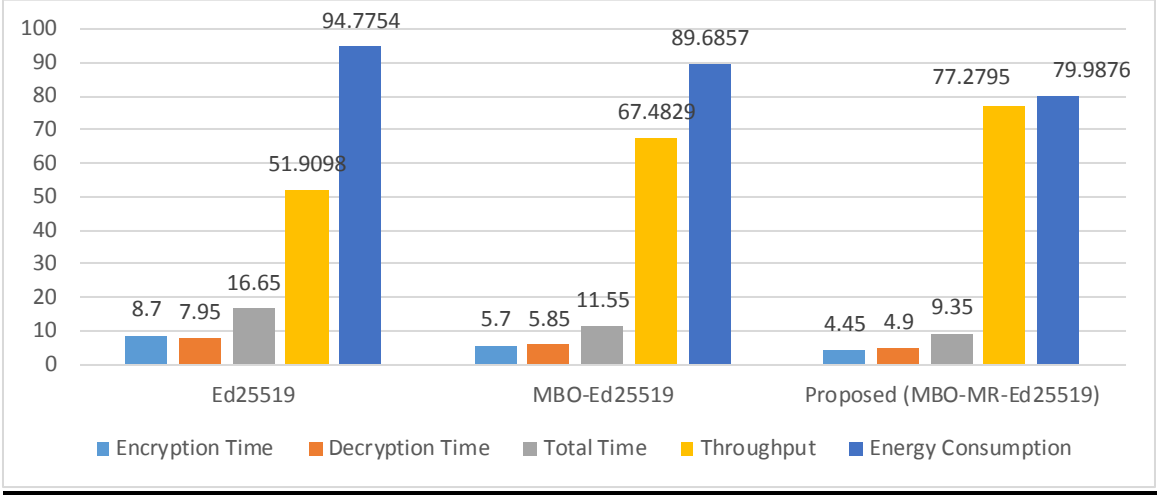

Figure 8: Performance Analysis of Graphical Representation

5. CONCLUTION

The Monarch Butterfly Optimization and MapReduce algorithms are based on Edwards25519. A proposed model (MBO-MR-Ed25519) has several advantages made possible by combining the benefits of existing algorithms for high complexity time choices. The optimization technique is used to generate optimal values from the data to select the most appropriate attributes. It is the best approach to handle colossal datasets, as it is an open-source programming structure that is dispersed. A session key is generated for signature generation and verification using the proposed method. In order to compress the data, the optimal value is determined based on Monarch Butterfly Optimization. The optimal value is then applied into MapReduce to create the compressing algorithm. To enhance data authentication security, the compressed data is applied to Ed25519 algorithm for key generation, signature generation, and signature verification. The proposed (MBO-MR-Ed25519) obtained the high performance better than (MBO-Ed25519). Among our model have overcome the limitation, with less time and increases the security. The performance of the proposed approach gives a significant execution of Total Time is 9.35(m). Our approach proved its efficiency in optimized parallel processing on the Cardiovascular Records dataset. Our model is chosen because of less complexity of high security.

REFERENCES

1. Thoyazan Sultan Algaradi and B Rama, “A Novel Blowfish Based-Algorithm to Improve Encryption Performance In Hadoop Using Mapreduce”, International Journal Of Scientific & Technology Research Volume 8, Issue 11, pp. 2074-2081, November 2019.
2. Zhou, Huan, et al. "A hybrid butterfly optimization algorithm for numerical optimization problems.", Computational Intelligence and Neuroscience, pp. 1-14, 2021.
3. Yi, Jiao-Hong, Jian Wang, and Gai-Ge Wang. "Using monarch butterfly optimization to solve the emergency vehicle routing problem with relief materials in sudden disasters.", Open Geosciences, Vol. 11, Issue. 1, pp. 391-413, 2019.

4. Nagarajan, G., and K. Sampath Kumar. "A Novel Monarch Butterfly Optimization with Attribute based Encryption for Secure Public Cloud Storage.", Vol. 12, Issue. 4, pp. 1044-1054, 2021.
5. Usmani, Mohammad A., et al. "Efficient PUF-based key generation in FPGAs using per-device configuration." IEEE Transactions on very large scale integration (VLSI) systems, Vol. 27, Issue. 2, pp. 364-375, 2018.
6. Pulgar-Rubio, F., et al. "MEFASD-BD: multi-objective evolutionary fuzzy algorithm for subgroup discovery in big data environments-a mapreduce solution." Knowledge-Based Systems, Vol. 1, Issue. 17, pp. 70-78, 2017.
7. Sowkuntla, Pandu, SravyaDunna, and PSVS Sai Prasad. "MapReduce based parallel attribute reduction in Incomplete Decision Systems." Knowledge-Based Systems, Vol. 21, Issue. 3, pp. 106677, 2021.
8. Dharavath, Ramesh, Samuel Nyakotey, and Damodar Reddy Edla. "MapReduce based integration of health hubs: a healthcare design approach." Health and Technology, Vol. 9, pp. 737-750, 2019.
9. Dilip, Golda. "An efficient privacy preserving on high-order heterogeneous data using fuzzy K-prototype clustering." Journal of Ambient Intelligence and Humanized Computing, Vol. 12, pp. 5191-5203, 2021.
10. Islam, MdMainul, et al. "FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field." IEEE Access, Vol. 7, pp. 178811-178826, 2019.
11. Gao, Lili, et al. "DPF-ECC: A framework for efficient ECC with double precision floating-point computing power." IEEE Transactions on Information Forensics and Security, Vol. 16, pp. 3988-4002, 2021.
12. Islam, MdMainul, et al. "Design and implementation of high-performance ECC processor with unified point addition on twisted Edwards curve." Sensors, Vol. 20, Issue. 18, pp. 5148, 2020.
13. Aggarwal, Shubhani, and Neeraj Kumar. "Digital signatures." Advances in Computers. Elsevier, Vol. 121, pp. 95-107, 2021.
14. This Cardiovascular Disease dataset is publicly available in Kaggle website, https://www.kaggle.com/sulianova/cardiovascular-disease-dataset.
15. C. Kavitha, S. R. Srividhya, Wen-Cheng Lai and Vinodhini Mani, "IMapC: Inner Mapping Combiner to enhance the performance of MapReduce in Hadoop." Electronics, Vol. 11, Issue. 10, pp. 1-16, 2022.
16. KhushbooKalia and Neeraj Gupta, "Analysis of hadoop MapReduce scheduling in heterogeneous environment" Ain Shams Engineering Journal, Vol. 12, Issue. 1, pp. 1101-1110, 2021.
17. Priyank Jain, ManasiGyanchandani and NilayKhare, "Enhanced secured map reduce layer for big data privacy and security." Journal of Big Data, Vol. 6. Issue. 1, pp. 1-17, 2019.
18. F.Ouatik, M.Erritali, and M. Jourhmane. "Comparative study of MapReduce classification algorithms for students orientation." Procedia Computer Science 170 (2020), pp. 1192-1197, April 2020.

19. Priyank Jain, ManasiGyanchandani and NilayKhare, "Enhanced secured map reduce layer for big data privacy and security." Journal of Big Data, Vol. 6. Issue. 1, pp. 1-17, 2019.
20. ViploveKadre and SushilChaturvedi, "AES – MR: A Novel Encryption Scheme for securing Data in HDFS Environment using MapReduce", International Journal of Computer Applications, Volume 129 – No.12, pp. 12-19, November 2015.
21. BinhKieu-Nguyen, Cuong Pham-Quoc, Ngoc-Thinh, Cong-Kha Pham and Trong- Thuc Hoang, "Low-Cost Area -Efficient FPGA-based Multi-Functional ECDSA/EdDSA", Cryptography, Vol. 3, Issue. 2, pp. 25, May 2022.
22. M.J. Bharathi, V.N. Rajavarman and R. Shobarani, "Implementation of Digital Signature Algorithm using Big Data Sensing Environment", International Journal of Engineering and Advanced Technology (IJEAT), Vol. 9 Issue. 2, pp. 4128-4131, December, 2019.
23. Shankar, Gauri, et al. "Improved Multisignature Scheme for Authenticity of Digital Document in Digital Forensics Using Edward-Curve Digital Signature Algorithm.", Security and Communication Networks, pp. 1-18, 2023.
24. S.SyedNawas Husain and Dr.R.Balasubramanian, "Comparative Analysis of Standard Elliptic Curves Based Elliptic Curve Cryptography for Secured Students Academic Records" NeuroQuantology, Vol. 20, Issue. 15, Page no. 7733-7749, November 2022.
25. S.SyedNawas Husain and R.Balasubramanian, "An Efficient (MBO-Ed25519) Algorithm for Securing Student Academic Records", European Chemical Bulletin, Vol. 12, Issue. 6, pp. 2371 – 2386, 2023.