Journal of Medical Systems (2019) 43: 124
https://doi.org/10.1007/s10916-019-1250-4

TRANSACTIONAL PROCESSING SYSTEMS

IFHDS: Intelligent Framework for Securing Healthcare BigData

Youssef M. Essa[1] · Ezz El-Din Hemdan[2] · Ahmed El-Mahalawy[2] · Gamal Attiya[2] · Ayman El-Sayed[2]

Received: 25 November 2018 / Accepted: 15 March 2019 / Published online: 27 March 2019

Abstract

Big data has become one of the most imperative technologies for collecting, handling and analysing enormous volumes of data in a high-performance environment. Enterprise healthcare organizations needs high compute power for the large volume of sensitive data, as well as large storage for storing both data and results, preferably in the cloud. However, security and privacy of patient data have become a critical issue that restricts many healthcare services from using cloud services to their optimal level. Therefore, this issue has limited healthcare organizations from migrating patient data to a cloud storage, because the cloud operators have chance to access sensitive data without the owner's permission. This paper proposes an intelligent security system called Intelligent Framework for Healthcare Data Security (IFHDS). IFHDS enables to secure and process large-scale data using column-based approach with less impact on the performance of data processing. The intelligent framework intends masking personal data and to encrypt sensitive data only. The proposed IFHDS splits sensitive data into multiple parts according to sensitivity level, where each part is stored separately over distributed cloud storage. Splitting data based on sensitivity level prevents cloud provider to break complete record of data if succeeds to decrypt part of data. The experimental results confirm that the proposed system secure the sensitive patient data with an acceptable computation time compared to recent security approaches.

Keywords Big data · Cloud computing · Healthcare · Security

Introduction

In recent decades, the size of data digitization in healthcare has increased for different data varieties such as structured, semi-structured, and unstructured data [1, 18]. The healthcare industry is facing multiple challenges. There are cost of care delivery, data volume growth in relation to patients, doctors, and healthcare entities. Healthcare is one of the sectors generating huge, velocity, and a vast variety of data [2, 12, 13]. Consequently, to obtain high performance in data processing, the healthcare entities are moving to use big data technology [25, 26].

Apache hadoop and Apache spark [9, 16] are the most common big data processing platforms used today over distributed systems. The goal is to process data among hundreds or thousands of machines in a cluster. They provide fault tolerance, and high abstraction in building applications using the MapReduce processing model [14]. In the healthcare sector, the big data solutions offer to process and analyse this huge amount of data. Thus, the big data platforms help to gain more information, predict outcomes for patients by getting a 360-degree view of the patient data, and real-time decisions making [11].

Therefore, the healthcare organizations are moving toward using cloud computing. This is save costs, improve care management, analyse different types of data and manage healthcare services [3]. Moreover, the cloud providers use Mass Distributed Storage (MDS) to scale up the data storage size and provide high performance data retrieval [34]. Basically, the main idea of processing big healthcare data on the cloud is using hadoop and spark to split data into sub-files and store them on the cloud [35, 36]. Indeed, storing health data on the cloud provides several advantages to healthcare organizations. For instance, bandwidth for data transferring, increased accessibility so data can be accessed from anywhere, and cost savings due to reduced annual operating costs. Big data services based on cloud computing are one of the most attractive research areas in the field [4, 5].

This article is part of the Topical Collection on *Transactional Processing Systems*

✉ Youssef M. Essa
yosufessa@gmail.com

[1] Senior Big Data Engineer, Idealo, Berlin, Germany

[2] Faculty of Electronic Engineering, Menoufia University, Menoufia, Egypt

The reason behind that is that healthcare data contains sensitive information and security is the biggest issue for healthcare organizations. This issue needs a solution to protect healthcare data and mask personal data from cloud providers [6, 36, 41, 43]. In recent years, there are many studies carried out to protect sensitive data by masking personal data. However, based on the HIPAA model, masking data is still readable to anyone and is not enough for healthcare organization. Healthcare entities requirements need to store sensitive data as a cipher text [44].

Data can be secured by using various techniques such as authentication, integrity, and encryption. Data encryption is nothing but change data into secret text using encryption algorithms. Also, it means calculations that transform plain text into cipher text, a form that is non-readable to unauthorized parties. The encryption process of data can be done at sender's side and data decryption is done at receiver's side. The encryption algorithms can be classified into two main categories, namely, symmetric key cryptography and asymmetric key cryptography. On the one hand, symmetric-key algorithms use the same key for both encryption and decryption. For instance, Data Encryption Standard (DES), Advanced Encryption Standard (AES) and blowfish. On the other hand, asymmetric key algorithms utilize different keys for encryption and decryption. For example, Rivest-Shamir-Adleman (RSA) and Elliptic Curve Cryptography (ECC).

Security and privacy threats to health data stored in the cloud are mainly categorized into two areas. The first is the internal or external cloud provider agents abusing privileges to access healthcare data. This is an "organizational threat", it means that a cloud provider employee can provide confidential data to outsiders. The second area is "systemic threats", which occurs during the flow of healthcare data transmission. This could be in the form of deleting, modifying, denying or updating patient records during the transmission process [6, 37].

The current most common active approaches to protect healthcare data is applying encryption algorithms on the whole raw data using one key. Encrypt data uses only one key affects the performance, especially when many columns not storing sensitive data. So, these approaches cannot meet healthcare organizations requirements because the cloud providers still able to decrypt the health data the on-cloud storage. Thus, it is essential to solve security problems with less impact on the performance. This paper proposes an efficient IFHDS approach using encryption algorithms in a distributed manner among different cloud platforms. IFHDS protects only the sensitive data with less impact on performance and latency.

IFHDS enhances the performance for health data processing by encrypting each column based on the security needs instead of working on raw mode. The platform processes and encrypts different levels of plain text. Also, IFHDS encrypts data on different levels and provides different key for each column based on the security level. So, the proposed approach increases level of security to protect data through isolating any malicious activities to access patient's data.

Figure 1 illustrates the architecture of proposed IFHDS framework. The broken-arrow lines represent the client perspective and the solid ones represent that of the cloud providers. As shown in Fig. 1, to optimize cluster resources, instead of encrypting all data, the first step defines the sensitivity level of the data and splits it into two types of files. The first type contains sensitive data and the second contains non-sensitive data as a plain text. The first step is mainly supported by the Data Sensitivity Level algorithm (DSL). This algorithm defines the sensitivity level for each attribute and separate sensitive data from non-sensitive data. Then, the Sensitivity Level Encryption (SLE) algorithm is developed to use a cryptographic algorithm regarding sensitivity Level. To perform replication and distribution of encrypted data over different cloud providers the Duplication and Distribution Data Security (D3S) algorithm is defined to support this step. Decrypt Sensitivity Data (DSD) is responsible for decryption processing in retrieval data. Hence, the main problem solved by IFHDS is protecting healthcare data from being directly reached by cloud providers.

The rest of this paper is structured as follows: Section 2 provides related work in this innovation area. Section 3 introduces the proposed system of experimental study, and a results analysis and discussion are introduced in Section 4. The paper conclusion and future work are provided in Section 5.

Related work

Big healthcare data is beyond traditional data processing in terms of performance, storage, and manner of analytics in order to deliver healthcare entities requirements. Therefore, it creates challenges in data storage, and security [20]. For encrypting small amounts of plain data, the current encryption algorithms are a possible solution using a single machine. However, to secure big health data without improper performance, the industry needs innovative approaches based on distributed storage and processing on the cloud computing [2, 21].

In recent years many studies have developed new approaches to protecting health data by tracing workflow of data execution on the cloud using multi-agent computing [30, 38]. In [47], the authors proposed a new platform called a Fully Homomorphic Encryption for Blend Operations (FHE-BO). For arithmetic operation, FHE-BO proved that had an acceptable error caused by the bitwise operations. Generating encryption matrices' time was random. This time depends on the efficiency of creating invertible matrices. FHE-BO is powerful approach when use it in basic arithmetic operations. However, there is no practicable application yet on this approach due to is hard to achieve correct result through querying cipher text.

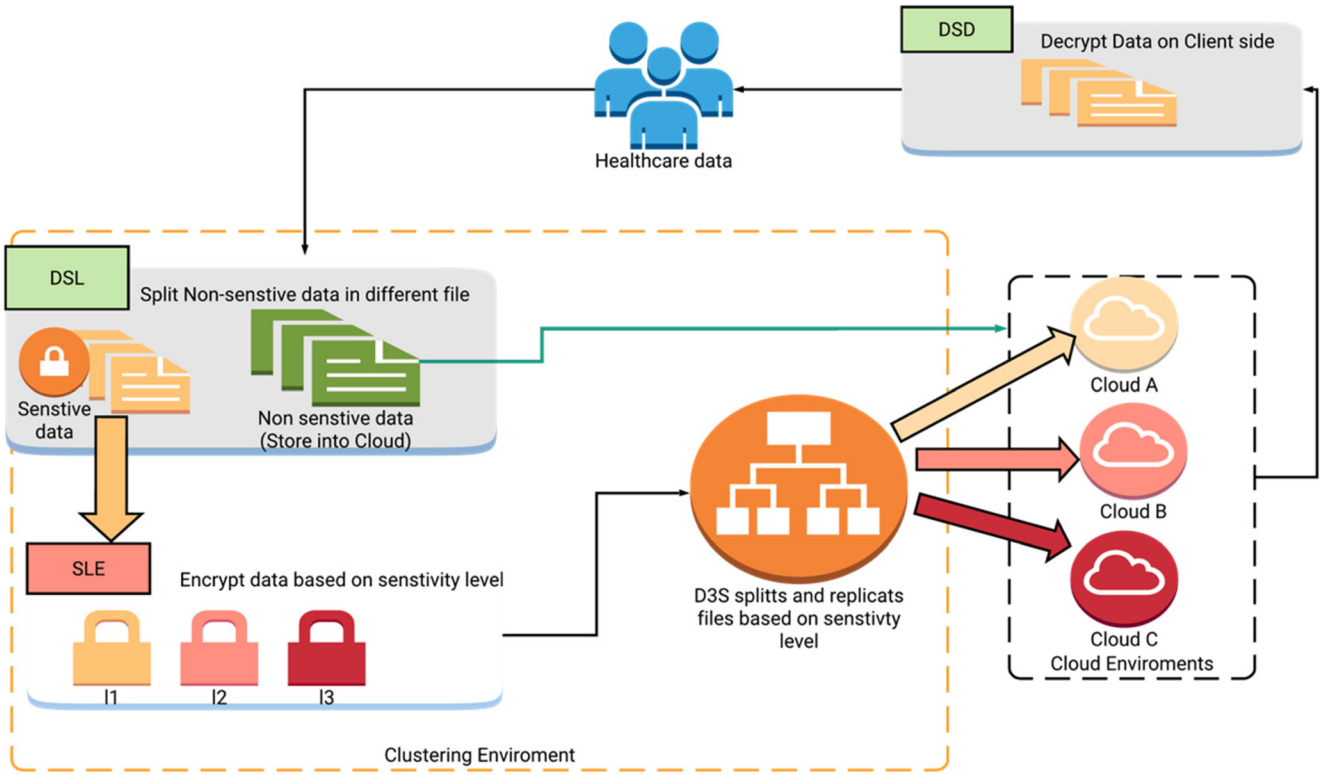

Fig. 1 Proposed intelligent framework for healthcare data security

Other studies focus on access control mechanisms to authenticate accessing of the data [39], whilst other research focus on encrypting the whole data using encryption algorithms [10, 15, 17]. The authors [22] have introduced a new technique called Computing on Masked Data. This approach allows computations on masked data instead of traditional cryptographic techniques that make overheads when performing computations on big data.

In [23], a framework is proposed for securing sensitive big data on the cloud. The data is secured throughout delivery, storage and destination using heterogeneous proxy re-encryption algorithm. However, from a performance perspective, this framework needs an optimization in heterogeneous proxy re-encryption algorithm, to improve performance while applying encryption algorithms to big data. In paper [24] a secure electronic healthcare system is introduced that protects patient confidentiality. Also, this system is able to centralize the collection of patient data to prevent any unauthorized access. Other recent research is starting to focus on using multiple cloud providers to secure sensitive data like [8].

In summary, recent studies focus on solving the problem of sensitive data abuse on cloud providers using two approaches. The first approach is restricting employee behaviours using General Data Protection Regulation (GDPR) and most companies apply this type of regulation. The second approach is the common one which protects data using encryption approaches, like ABE and RSA. However, this type of data security does not meet the requirements of healthcare organizations. The performance issues caused by encrypting the whole data. Another reason is that the cloud providers can leak information if sensitive data is successfully decrypted because most frameworks depend on one encryption key to encrypt all data and store all the data in one place. Therefore, IFHDS is designed for big healthcare data processing over distributed storage on the cloud. IFHDS enabled data to be secure, reliable and available in cloud storage.

Proposed intelligent framework for healthcare data security

As healthcare organizations look to move into the cloud computing, many organizations are implementing new security measures and performance optimizations [27–29, 42]. The storing of immense amounts of data using traditional encrypt/decrypt, mechanisms are not sufficient for two main reasons [19]. The first reason is that, the recent approaches [22–24] are used one key to encrypt all data. Encrypting all data using a single key is risky if the cloud providers succeed to generate decryption key to access patient's data.

Therefore, the need to invent new ways of data processing/classification is greater than ever to reduce the overheads while applying security algorithms. In this ever-changing landscape, this paper presents the comprehensive approaches

for data security, performance, reliability and flexibility. This is done by proposing a data sensitivity-based model to govern the data classification and tiers using DSL and SLE algorithms. As shown in Fig. 2, the proposed IFHDS framework consists of four algorithms as following:

DSL algorithm

Spark masks personal data based on GDPR requirements for patients and doctors using hashing table which stored in EHR. DSL received masked data to start encryption process. Encryption frameworks such as SA-EDS encrypts the whole source data while distributing data among cloud providers and induce computational overheads and additional latency. Hence our proposed DSL algorithm algorithm splits masking data into sensitive data or plain data. Then, DSL creates new files with prefix <sensitive> or < plain> depending on the sensitivity level of the data as shown in algorithm 1. This means, the masking data will split into two types of files, one including the sensitive data and the other including the plain text. This algorithm optimizes the cluster resources by encrypting only the sensitive data while storing or retrieving huge data. In the same time of encrypting sensitive data, DSL sends plain data to cloud storage.

In algorithm1, the data owner adds label L for each attribute to define security level. As shown in Fig. 2, all security levels defined by owner saved in yaml configuration file which used in splitting data. So, based on the security levels in yaml file, the security tiers can be classified into four security level as follows:

- **Security Level 0 (plain data):** Aggregation information like total income, outcome or the total number of patients per healthcare entity.
- **Security Level 1:** In this level, the data is plain text, such as address, telephone, Fax for the hospital.
- **Security Level 2:** A dataset that includes data of disease, healthcare entities and financial information.
- **Security Level 3:** Healthcare dataset including all private data of patients and doctors.

The DSL algorithm stores attribute E with a security level (SL) greater than 0 with prefix "sensitive" file and uses prefix "normal" on plain text with security level 0. This flow done through spark environment. Spark read all data in memory as single data frame called "main_df" to keep all processing in memory. DSL creates two dataframes from "main_df". The first dataframe called

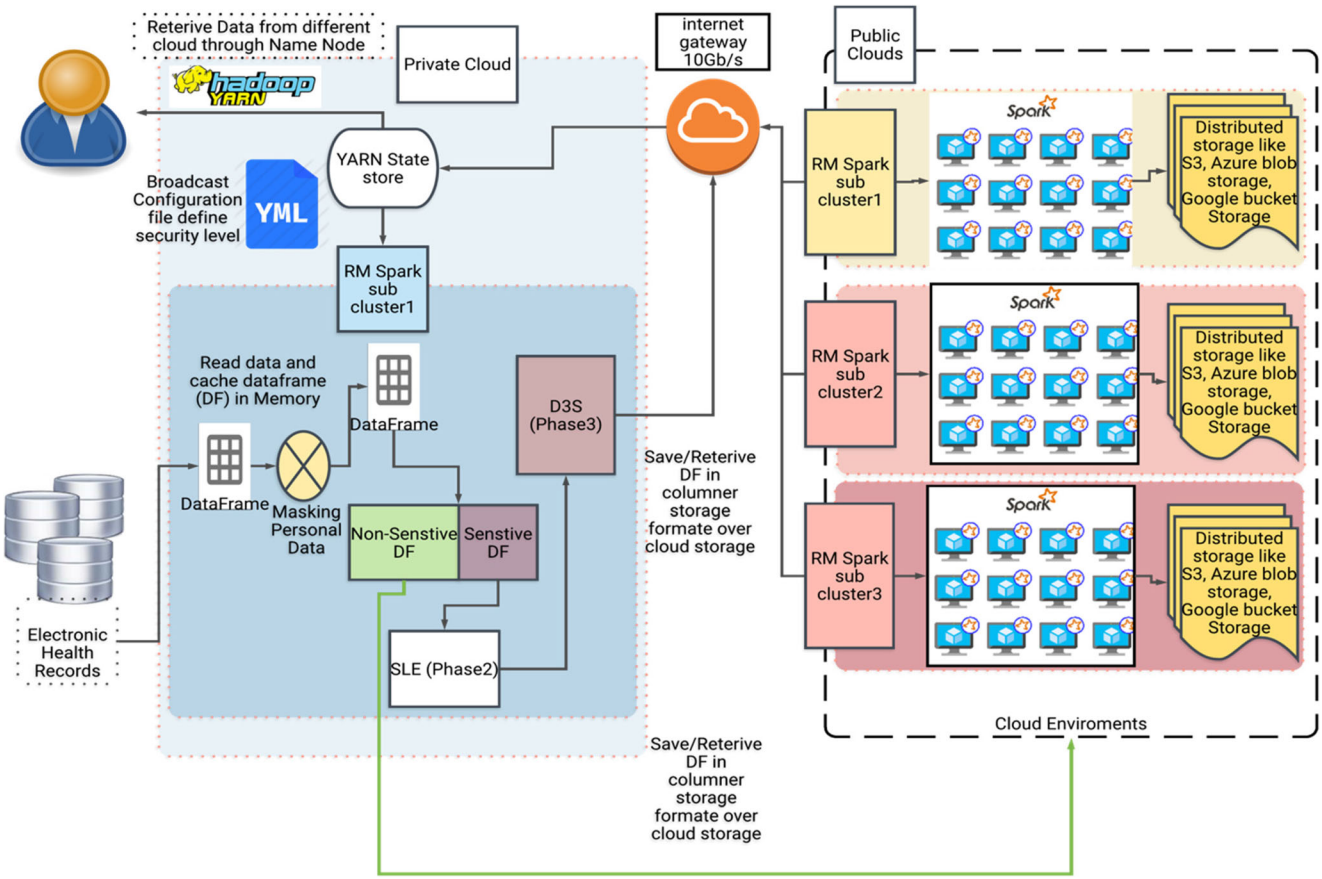

Fig. 2 IFHDS system architecture

"non_senstive_df" contains all fields with security level 0 to store it in cloud environment. The second dataframe called "senstive_df" contain the rest of data that need encryption. Finally, to clean memory the DSL un-persist main_df from the memory and caching non_senstive_df to use it in next step.

The advantage of DSL algorithm is breaking the dependency between sensitive data and normal data. This advantage saves a time and cost by sending normal data into cloud storage without encryption processing. This means the transmission of data will be faster due to two operations being performed in parallel. The first operation is transferring plaintext data into distributed storage on the cloud provider. The second one is starting encryption process of sensitive data in SLE algorithm.

Algorithm 1 DSL to split sensitive data from normal data

Requires: security labels S(L).
Ensures: User defined code will be executed for each attribute of E ∈S(L).

```
1. Security labels for attribute S(L)        ◅ Start the algorithm
2.  procedure splitting data (E)
3.  for i ← 1, i<= number_of_E do
4. if L> 0 then
5. <file_name>_senstive ← capture E from file.
6.        else
7. <file_name>_normal ← capture E from file.
8. end if
9. END Loop
10. Send <file_name>_normal to cloud storage.
11. end procedure
```

SLE algorithm

SLE encrypts the data based on the security level for each attribute before loading a sensitive data into a cloud storage. To evaluate our approach, we are using three cryptographic algorithms; Advanced Encryption Standard (AES), Data Encryption Standard (DES) and Blowfish [7]. IFHDS also categorizes these algorithms based on security strength, as is already defined in reference [33]. However, for more flexibility, the data owner can change the cryptographic algorithm for each level. So, by default AES algorithm used in level 3, Blowfish in level 2, and DES in Level 1.

The idea of ranking sensitive data is to avoid producing a breakpoint by using only one encryption key for sensitive data. Also, from a performance perspective, it is important to meet all healthcare organization security requirements without reducing performance of data processing. So, the best security approach is to encrypt data based on security level for each attribute before loading it into cloud storage. This is will meet both security and performance requirements for each level. To do so, as shown in algorithm 2, SLE starts to read security level for each attribute and encrypt data using the appropriate cryptographic algorithm for this level. To enhance the performance of the encryption process, we are using Apache Spark as distributed computing system over Yet Another Resource Negotiator (YARN) [31]. SLE encrypts all attributes in parallel instead of using sequential process. D3S responsible to load encrypted data from spark into cloud providers.

SLE algorithm enhances the performance of encrypting data using two strategies. The first strategy uses ranking of sensitivity to load balance between the strength of security algorithm and performance of encryption. The second strategy is encrypting all attributes at the same time using big data processing technology like spark to achieve the highest performance. SLE caches senstive_df to split each level as a new data frame to encrypt it with appropriate security algorithm. Then, send all encrypted data frames to D3S algorithm.

Algorithm 2 SLE to encrypt data over spark cluster

Requires: security labels S(L).
Ensures: User defined code will be executed for each attribute of E ∈S(L).

```
1. Initiate spark cluster based on YARN
2. procedure reading (E) from<file_name>_sensitive in parallel
   using spark.
3.  for i ← 1, i<= number_of_Edo
4. if L = 1 then
5.  Overwrite <file_name>_sensitive ← Encrypting E using DES.
6. Elif L=2
7.  Overwrite <file_name>_sensitive ← Encrypting E using
   Blowfish.
8. Elif L=3
9.  Overwrite <file_name>_sensitive ← Encrypting E using AES.
10.       else
11. Print "level not valid".
12. end if
13. END Loop
14. Send <file_name>_ sensitivety D3S algorithm.
15. end procedure
16. Terminate spark cluster
```

D3S algorithm

Instead of keeping data in the hand of only one cloud provider, D3S works on splitting data into different cloud providers as each cloud provider would have incomplete data. This strategy will keep health data secure even if the cloud provider succeeds to decrypt the data. Indeed, Security-Aware Efficient Distributed Storage algorithm (SA-EDS) approach only supports splitting data over two cloud providers. Therefore, failing to provide flexibility or reliability of health data when the cloud service goes down. Reliability of a distributed storage in cloud computing may be defined as the probability that the client can retrieve data successfully [9, 30]. However, successful retrieval of data from cloud providers depends on the reliability of distributed storage in the cloud.

Hence, to successfully keep health data available 24/7 only one main condition needs to be satisfied, being that the storage services in the cloud must be operational all the time. Although this may sound simple it is a very difficult condition from a practical point of view. Health data is critical data healthcare organizations need to keep all data available at all

times. IFHDS resolves this issue using D3S algorithm which considers each cloud provider as a data node in hadoop cluster over YARN as shown in Fig. 4. Apache hadoop provides reliability by replicating each bulk of data into three different nodes [31, 40]. In the case that one of the nodes (cloud provider) goes down, the data owner in the healthcare organization can retrieve data from the replicating node. Apache hadoop also provides more flexibility to add any number of nodes in runtime and automatically balance further data over different nodes.l

The idea of the D3S algorithm is reading encrypted data from SLE algorithm and using hadoop cluster to split and replicate data as shown in algorithm 3. D3S is a column-based algorithm. It means storing each column as a bulk and replicating this bulk over cloud providers. Also, in D3S, the data owner can define replication factor (RF) which represent the replication number for each bulk.

Algorithm 3 D3S to Split data into bulk and replicate it over cloud providers

Requires: <file_name>_sensitive (File), replication factor (RF)
Ensures: User defined code will be executed for each attribute of E ∈File.

1. Initiate hadoop cluster based on YARN
2. procedure reading (E) from File in parallel using hadoop. ◁Splitting step
3. for i ← 1, i<= number_of_E do
4. Replicate E into cloud providers (nodes) with RF
5. Resource Manager updating the map of data bulks
6. END Loop
7. end procedure

DSD algorithm

The main task of DSD algorithm is enabling the data owner in the healthcare organization to retrieve data from different cloud providers. The data owner requests data from the resource manager in hadoop cluster. Resource manager has access to mapping table of data bulks among different cloud providers or data nodes. DSD decrypts data via appropriate decryption key for each attribute based on the security level that is defined in SLE algorithm. All workers in the Yarn subclusters able to use the decryption key through hadoop KMS [43, 44]. KMS used to manage the keys in hadoop environment. We used spark over hadoop as a distributed environment to provides in memory processing and easy to distribute encryption keys over worker machines. However, as shown in Fig. 3 KMS enhanced to encrypt data on column based instead of raw. So, the Master node generates key for each security level and store all keys in KMS to use it later when end user query stored data. Also, DSD algorithm serve client queries by distributed required keys related to the query over workers in the cluster.

All clusters running over YARN Federation [45, 46]. So, each could environment used own Resource Manager (RA) to manage the data processing inside the spark cluster on the same cloud. This approach scales a single YARN cluster to tens of thousands of nodes, by federating multiple YARN subclusters. The proposed approach is to divide a large cluster and

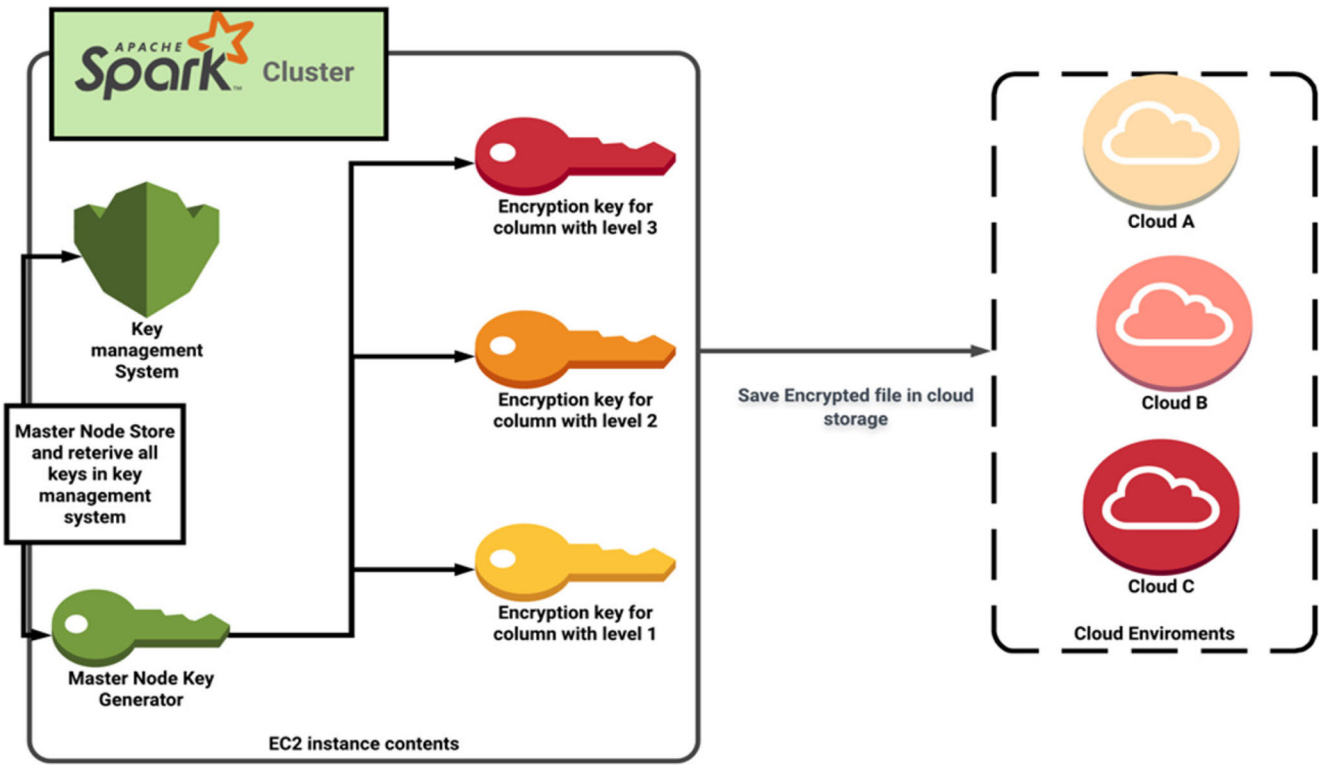

Fig. 3 Key management system

Table 1 Software and hardware cluster components

Attribute	Value
Number of Nodes	21
Memory Size (GB)	500
Network Bandwidth	40GB inside cluster/ 10Gb over Internet to public cloud
Number of CPUs/Node	96
CPU	Core i7 2.3 GHz
YARN Memory	470
Big data Ecosystems	spark, HDFS, MapReduce
Operating System (OS)	Unix
Big Data Platform	Cloudera
Replication Factor	1

spread data into smaller units called sub-clusters. Each one of the sub-cluster work on different cloud environment with its own YARN RM and compute. From application view, the federation combine all sub-clusters together as one large YARN cluster to the applications. YARN federation will be able to schedule tasks on any node of the federated cluster. Algorithm 4 shows the pseudo code of DSD using the following procedure:

1. Client request Data from hadoop.
2. Resource Manager defines bulk locations in different data nodes (cloud providers).
3. Data Nodes send data to DSD in parallel.
4. DSD reads security level for each attribute to know which security algorithm is used to encrypt attribute.
5. DSD decrypts each attribute using the same algorithm used in encryption.

After DSD finishes, hadoop stores decrypted data in output directory which is configured by client.

Algorithm 4 Retrieve and decrypt data

Requires: Request data (one or more attributes) E
Ensures: User defined code will be decrypt each attribute of E ∈number of E.

```
1. Request number of E from hadoop
2. Resource manager allocates all bulks related to E from Data Nodes.
3. Data Node sends all bulks to DSD algorithm.
4. DSD starting to decrypt E in file using S(L) which defined in algorithm1.
5.  for i ← 1, i<= number_of_E do
6. if L = 0 then
7. <File>← Keep data as it is.
8. if L = 1 then
9.  Overwrite E on the <File> ← decrypt E using DES.
10. Elif L=2
11. Overwrite E on the <File> ← decrypt E using Blowfish.
12.  Else
13. Overwrite E on the <File> ← decrypt E using AES.
14. end if
15. END Loop
16. END DSD
17. hadoop saves data in HDFS.
18. Close hadoop Session
```

Table 2 Measure dates

Column name	Data Type	Security Level
Memo Measure_Name	Char(159)	L3
Measure_ID	Char(45)	L3
Measure_Start_Quarter	Char(8)	L1
Date Measure_Start_Date	Char(21)	L1
Measure_End_Quarter	Char(8)	L0
Date Measure_End_Date	Char(21)	L0

Experimental study

This section presents the experimental environment and the results of the practical experiments. The experiments are done to evaluate the proposed framework on different volumes of data and different security levels. It is then compared with the last approach used to encrypt data using distributed storage SA-EDS [8].

Experimental environment

The experimental environment for testing and evaluating the proposed system consists of hardware and software components for a cluster with specifications in Table 1. Hadoop cluster runs over heterogeneous machines that have different hardware and software specifications. Hadoop cluster consists of 50 machines connected with each other through a Local Area Network (LAN) in on-premise data center and uses 10 Gb/s internet connection to public cloud. The replication factor in IFHDS hadoop cluster set to 1 to maximize data storing performance and saving cost. The replication of huge data increases the storage cost. Currently, most of distributed storage uses a 3- replicas data strategy by default to provide data reliability. For instance, to store one gigabyte of data would need three gigabytes of storage. This storage affects the cost effectiveness of the cloud storage by adding two times additional cost. However, we set replication factor in

Table 3 Hospital general information

Column name	Data Type	Security Level
Provider ID	Char(8)	L3
Hospital Name	Char(52)	L0
Address	Char(52)	L1
City	Char(52)	L1
State	Char(4)	L1
ZIP Code	Char(7)	L1
County Name	Char(22)	L1
Phone Number	Char(12)	L0
Hospital Type	Char(38)	L0
Hospital Ownership	Char(45)	L2
Emergency Services	Char(5)	L1

Table 4 Patient laboratory data

Column name	Data Type	Security Level
Provider ID	Text	L2
Hospital Name	Text	L0
Patient ID	Text	L1
White blood cell count	Number	L3
Red blood cell count	Number	L3
Hemoglobin	Number	L3
Mean corpuscular volume	Number	L3
Platelet count	Number	L2
Lymphocytes	Number	L2
Basophils	Number	L2
Chloride	Number	L3
Carbon dioxide	Number	L3
Anion gap	Number	L3
Glucose	Number	L2
Blood urea nitrogen	Number	L2
Creatinine	Number	L2

hadoop cluster into 1, because the storage in cloud storage systems like Amazon S3 perform well in general. In Amazon S3, the average failure rate of a single storage unit is 99.99% over a year. This means that, the average annual loss rate of data stored without replication is 0.01% [50]. So, from cost and performance perspective, to keep the replication factor equal 1 is the best choice to IFHDS framework. All personal data generated randomly and IFHDS used two data sources for testing as following:

1) For Table 2 and Table 3, The open source database provided by Centers for Medicare & Medicaid Services [48].
2) Data in Table 4 is based on Chemical Effects in Biological Systems (CEBS) database [49].

Experimental comparison and discussion

In this section performance evaluation is carried out by measuring the lapsed time, CPU utilization, Yarn memory utilization, cluster load, and network usage for various quantities of data. It also, presents a comparison study between different sensitivity levels of data based on previous measurement factors against SA-EDS Fig. 4.

So, before comparing IFHDS with SA-EDS, as shown in Fig. 5 this paper presents a comparison between encryption algorithms that used by IFHDS and SA-EDS. The complexity of algorithms is analysed on the basis of memory, CPU, network, cluster usage, and encryption performance. Figure 5 illustrates the results for each factor. The experiments is done using 500GB of data.

The memory usage can be defined as the memory usage during storing and processing data in memory. Less memory usage will be the greater efficiency. The performance can

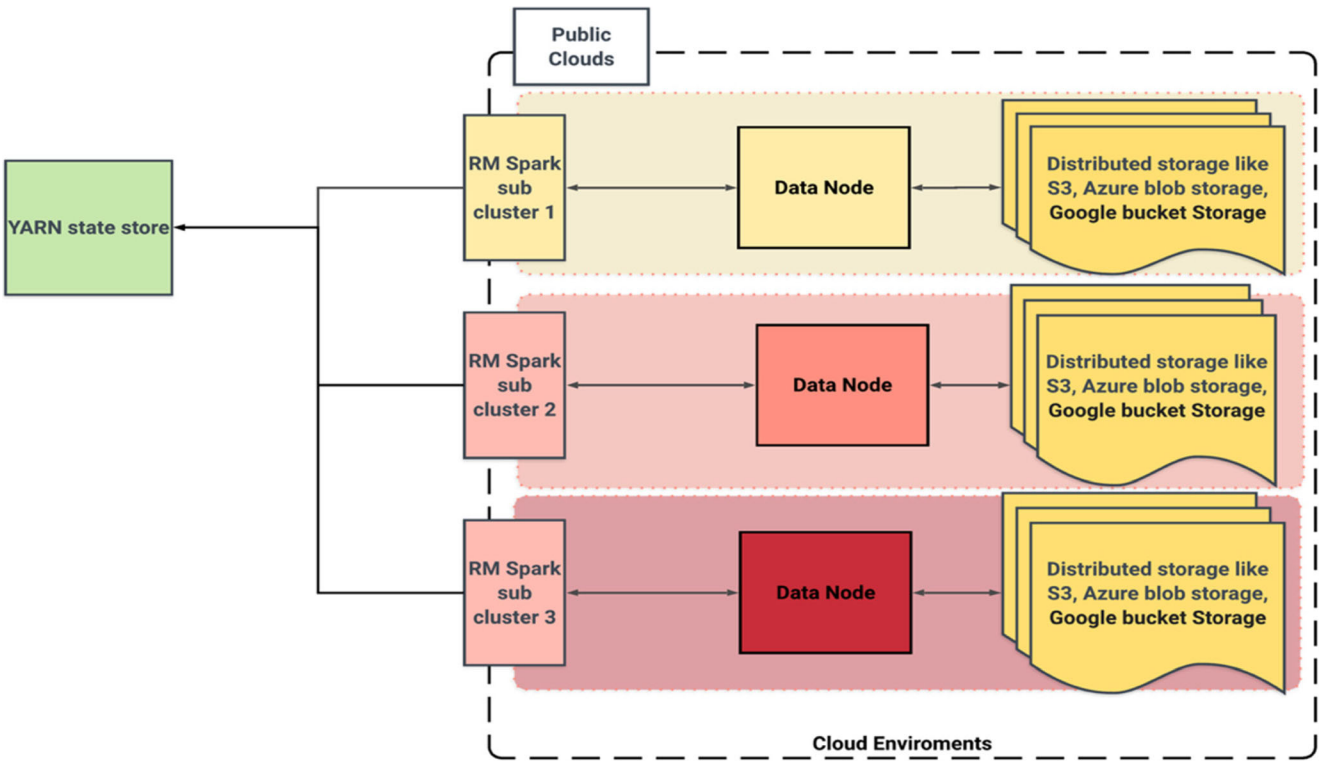

Fig. 4 Hadoop architecture for data replication among cloud providers

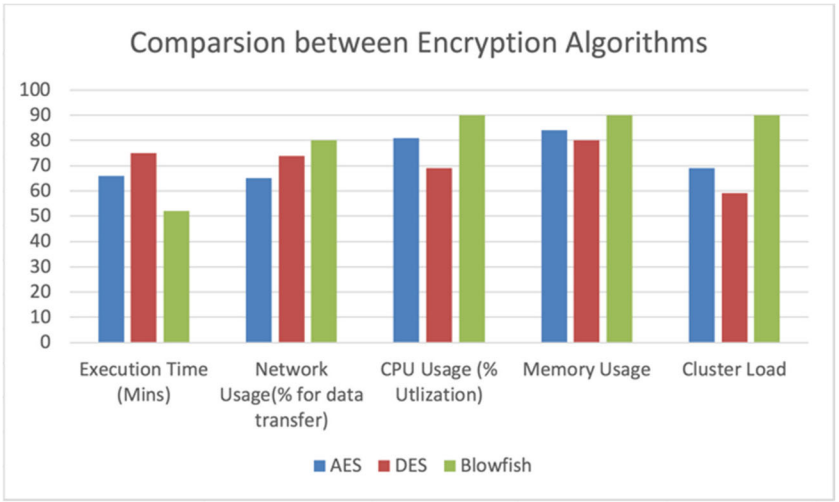

Fig. 5 Comparison between encryption algorithms

define as the execution time required by the algorithm to encrypt size of data. Network usage is representing the amount of data transferred between workers over clusters during encryption process. Finally, the cluster load defined as the percentage of resources used by encryption algorithm during end to end process. So, the better algorithm must be utilizing CPU and consuming less resources from cluster to achieve highest execution time. Based on the previous factors, the compared study found the blowfish achieved best performance with less cluster loading.

So, using the same type of encryption algorithms, the paper presents comparison between IFHDS and SA-EDS [8]. SA-EDS is the latest algorithm in the same research area. Tables from 2 to 4 illustrate the schema of the different tables that have been used in the experimental test. The raw data is composed of 33 columns with different sensitivity levels as shown in Tables 2, 3, 4. In all tables, there are three attributes; column name, data type and security level. The data type defines the type of data in each column. The last column defining the sensitivity level of the data. Each level, from L1 to L3, uses its own security algorithm in the encryption process. L0 representing plain text for plain text data.

To evaluate our platform, we have applied different security algorithms used in security levels to encrypt health data. Figs. [5–9] quantify the execution time, Memory usage, CPU usage and network transformation after encrypting all sensitive data using SA-EDS and compared with IFHDS. SA-EDS used only one encryption algorithm to encrypt all sensitive data. IFHDS framework used one or more encryption algorithm based on the sensitivity data for each attribute.

The execution time is the time need to encrypt, transform, and decrypt health data using SA-EDS and IFHDS approaches. Figure 6 compared execution time between SA-EDS and proposed IFHDS. The execution time is based on security level which used encryption algorithms AES, DES, and Blowfish. The SA-EDS approach uses single encryption algorithm to secure the whole healthcare data. The Blowfish algorithm is the best performing algorithm, but it does not meet all security requirements. There are some attributes need to be more secure using better and stronger algorithm like AES. In this situation, only one security algorithm has to be used to encrypt the whole data. However, if SA-EDS uses a more secure algorithm like AES, it will definitely meet all security requirements in different levels. But encrypting whole

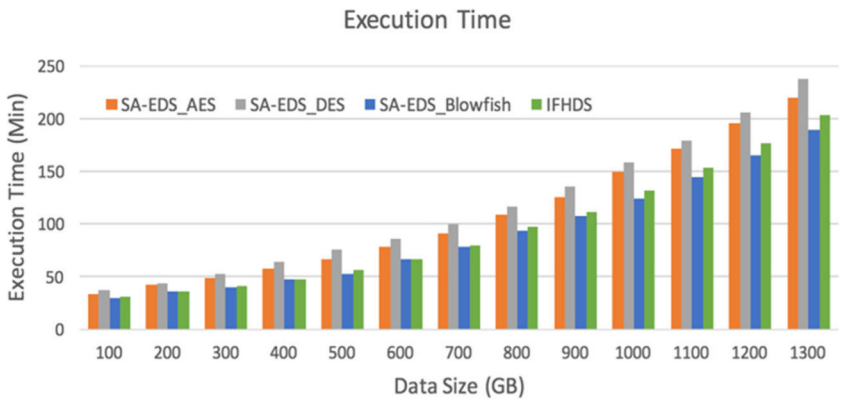

Fig. 6 IFHDS execution time comparing with various encryption algorithms using SA-EDS

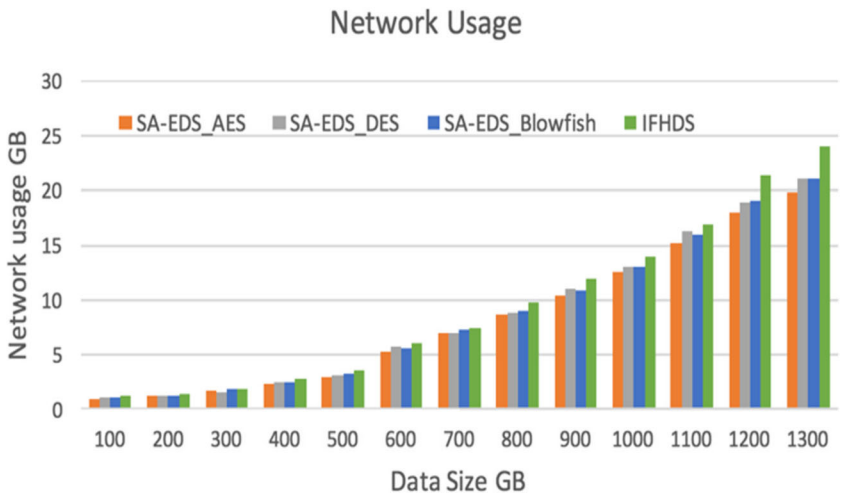

Fig. 7 IFHDS network usage comparing with SA-EDS

data using strong security algorithm, it will affect the performance requirements from healthcare organizations.

In contrast, our proposed algorithm IFHDS is designed to achieve the best performance and achieve all security needs. As shown in Fig. 6, the performance of IFHDS is better than all DES and AES to encrypt health data based on security level. This is because not all attributes need encryption due to DSL splitting normal data and sensitive data. Also, we should consider that the proposed IFHDS approach provides data replication for data availability. The replication process consumed time from the total execution time and this feature is not provided by the SA-EDS approach.

Figure 7, illustrates the network usage for each algorithm. To evaluate the network usage between workers in the cluster we are using Ambari tool [32]. We applied standard network tuning for 40 Gbps networking such as enabling TCP window scaling and increasing socket buffer sizes. Therefore, the platform used Abmari to measure the good throughput of a unidirectional bulk data transfer over a single TCP connection with standard 1500-byte MTU. The main restriction of IFHDS is that it needs strong network infrastructure. The IFHDS transfers packets to synchronize all operations between different nodes in hadoop cluster that verify the status of data nodes. The replication of data among cloud providers and associated heartbeat messages between hadoop nodes adds more network traffic.

Figure 8, illustrates the CPU usage in the end to end execution. IFHDS utilizes the CPUs to achieve high performance and meet security requirements. IFHDS uses different security algorithms based on the sensitivity level of data. Also, it able to encrypt multiple attributes in parallel because the encryption process is done through spark cluster.

In Fig. 9, the cluster load represents the average load on the cluster that was used by all consumers' components in this cluster. The consumer components mean the total number of nodes, the total number of CPUs, YARN memory, and the number of running processes. In small data below 800 GB, the average loading on the cluster does not vary greatly between different approaches. In small data size, the cluster resources are fast enough to finalize the processing of data efficiently. However, when processing data over 800 GB, it appears the SA-EDS_AES, SA-EDS_Blowfish, and SA-EDS_DES consume cluster components. This is affects the performance of algorithms. Therefore, as a direct result of the SA-EDS encrypting/

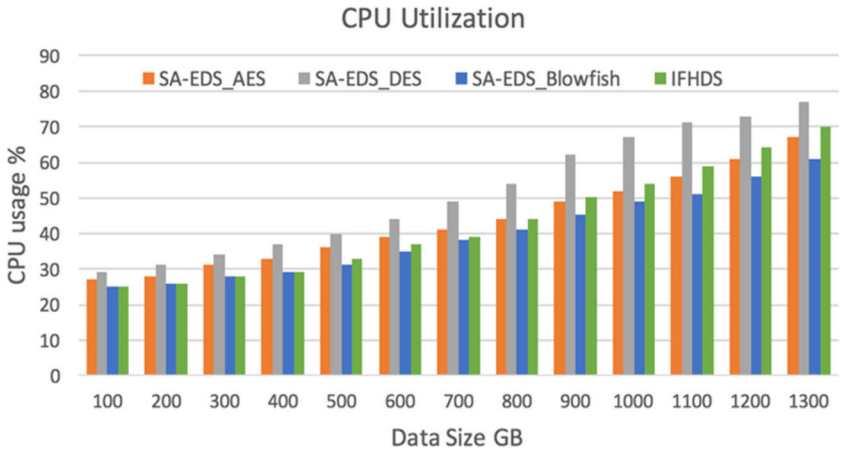

Fig. 8 IFHDS and SA-EDS CPUs utilization

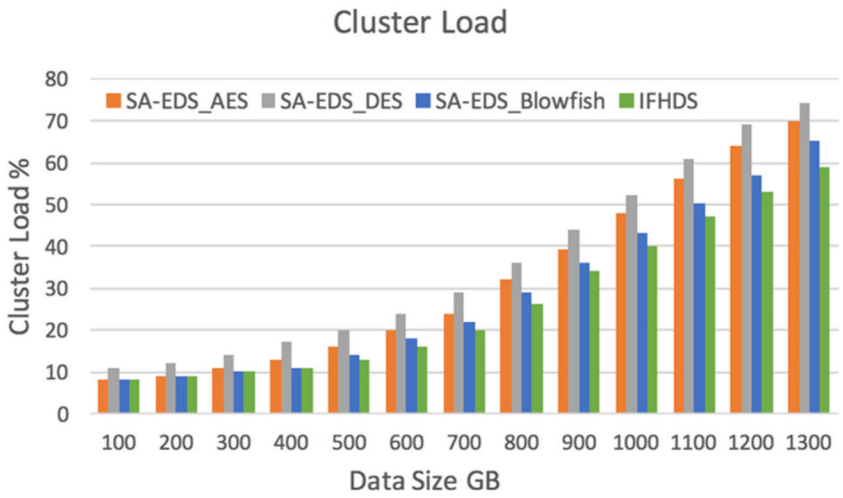

Fig. 9 Cluster load while using IFHDS and SA-EDS

decrypting data using sequential operations, it does not utilize cluster resources. For example, SA-EDS_AES, SA-EDS_Blowfish, and SA-EDS_DES keep YARN memory in an idle state between the CPU encrypting/decrypting the first batch of data and loading the second one.

Finally, to make sure the IFHDS framework working effectively over different cloud environments, IFHDS ran the complete process over YARN federation [45, 46]. End user submit spark application to Namenode which request resources from different RAs based on the availability of the resources in the cluster. Each RA send the status of the sub cluster into YARN state store machine which help main RA to define the free resource in each cluster. Then, YARN distribute the application into different RAs which each one of them responsible to execute one or more tasks from the job inside the cluster. At the end, the reducer in different sub clusters save the result in the target. So, from the result in Fig. 10 as expected we found the network effect the performance of the total execution time by around 2X slower over different geographical cloud environments. The network only effects the transferring result, not the actual processing execution time. However, using public clouds will add flexibility to run distributed job over different cloud platforms. This is will meet all security requirement from healthcare industry which can help in the future to enhance and optimize the data processing cross YARN federation.

In summary, the performance of the proposed IFHDS framework is evaluated and compared with various encryption algorithms such as AES, DES, and Blowfish based on SA-EDS approach. The simplest encryption algorithms like DES are a good choice to use in general projects that not have sensitive data. But most of healthcare data is sensitive. The performance of blowfish using SA-EDS approach is better than other algorithms when use it in lower level of sensitive data. However, in large and complex projects that need a more effective encryption algorithm, Blowfish is not the best choice. To solve this issue, the proposed IFHDS framework ranks sensitive data into levels before the encryption process. IFHDS selects the best encryption algorithm for each attribute and performs encryption and replication processes using hadoop cluster. The IFHDS approach uses hadoop cluster to enhance performance, utilizes hardware resources and meets the security requirements. Also, one of the main advantages of IFHDS is performing encrypt/decrypt operations among workers of big data cluster. In this case, the decryption process is faster due to it being executed through big data cluster not using a single machine.

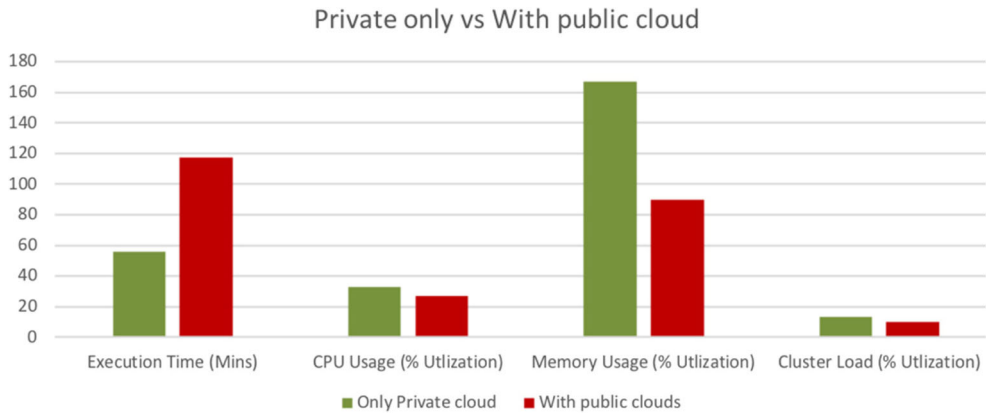

Fig. 10 IFHDS Performance comparison for data processing over public vs private clouds

Conclusion and future direction

This paper presented IFHDS as an Intelligent Framework to secure healthcare data among cloud providers without reducing efficiency. IFHDS performs encryption and decryption on the data based on big data cluster to improve performance while securing data. The IFHDS secured healthcare data using ranking cryptography approaches to minimize the processing time and to encrypt data based on the sensitivity level. From our experimental evaluation, the Blowfish encryption algorithm achieved high performance but doesn't enough to encrypt high-level sensitive data. AES algorithm provides high-security level but takes a lot of time to perform encryption and decryption operations.

On the other hand, the proposed IFHDS framework defines security levels for each attribute to achieve high performance using the most appropriate encryption algorithm for each level. IFHDS uses hadoop and spark to enhance encryption performance. Apache hadoop and spark enables IFHDS to encrypt and decrypt data and utilize hardware resources using parallel processing. IFHDS uses each cloud provider as a data node and provides data replication for data availability if one of the cloud providers goes down.

Nevertheless, the side effect of this approach is that the current version defines the sensitivity level of the data through a configuration file that is managed by the data owner. In future, we plan to develop a machine learning module that will classify healthcare data into different security levels. Also, we plan to enhance the network bandwidth to connect to public cloud providers through network bridge of multi-line direct connection.

Compliance with Ethical Standards

The authors declare that they have no of interest, there is no funding source, and this article does not contain any studies with human participants or animals performed by any of the authors. The authors whose names are listed conflict immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

1. Jiang, P., Winkley, J., Zhao, C., Munnoch, R., Min, G., and Yang, L. T., An intelligent information forwarder for healthcare big data systems with distributed wearable sensors. *IEEE Syst. J.* 10(3): 1147–1159, 2016.
2. Wang, Y., Kung, L., and Byrd, T.A., Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations. Technological Forecasting and Social Change, 2016.
3. Kohli, R., and Tan, S. S.-L., Electronic health records: how can IS researchers contribute to transforming healthcare? *Mis Quart.* 40(3):553–573, 2016.
4. Poppe, K., Wolfert, S., Verdouw, C., and Renwick, A European perspective on the economics of big data. *Farm Policy J.* 12(1): 11–19, 2015.
5. Wu, X., Zhu, X., Wu, G. Q., and Ding, Data mining with big data. *IEEE Trans Knowl. Data Eng.* 26(1):97–107, 2014.
6. Abbas, and Khan, A review on the state-of-the-art privacy-preserving approaches in the e-health clouds. *IEEE J. Biomed. Health Inform.* 18(4):1431–1441, 2014.
7. Singh, G., A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. *Int. J. Comput. Applic.* 67, no 19, 2013.
8. Li, Y., Gai, K., Qiu, L., Qiu, M., and Zhao, H., Intelligent cryptography approach for secure distributed big data storage in cloud computing. *Inform. Sci.* 387:103–115, 2017.
9. Essa, Y. M., Attiya, G., and El-Sayed, A., Mobile agent based new framework for improving big data analysis. IEEE international conference on cloud computing and big data (CloudCom-Asia 2013), FuZhou, China, 2013.
10. Pussewalage, G., Harsha, S., and Oleshchuk, V.A., A distributed multi-authority attribute based encryption scheme for secure sharing of personal health records. Proc 22nd ACM Sym Access Control Models Technol: 255–262. ACM, 2017.
11. Raghupathi, W., and Viju, Big data analytics in healthcare: promise and potential. *Health Inform. Sci. Syst.* 2(1):3, 2014.
12. Mathew, PrabhaSusy, and Anitha S. Pillai. "Big data solutions in healthcare: problems and perspectives. International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), IEEE, 1–6, 2015.
13. Priyanka, K., and NagarathnaKulennavar. "A survey on big data analytics in health care. *Int. J. Comput. Sci. Inform. Technol.*, vol 5, no. 4, pp. 5865–5868, 2014.
14. Dobre, C., and Xhafa, F., Parallel programming paradigms and frameworks in big data era. *Int J Parallel Program.* 42(5):710–738, 2014.
15. Xhafa, F., Li, J., Zhao, G., Li, J., Chen, X., and Wong, D. S., Designing cloud-based electronic health record system with attribute-based encryption. *Multimed Tools Applic* 74(10):3441–3458, 2015.
16. Zaharia, M., Xin, R. S. et al., Apache Spark: a unified engine for big data processing. *Commun. ACM* 59(11):56–65, 2016.
17. Liu, J., Huang, X., and Liu, J. K., Secure sharing of personal health records in cloud computing: ciphertext-policy attribute-based signcryption. *Fut. Gen. Comput. Syst.* 52:67–76, 2015.
18. Barbierato, E., Gribaudo, M., and Iacono, M., Performance evaluation of NoSQL big-data applications using multi-formalism models. Future Generation Computer Systems 37:345–353, 2014.
19. Cash, D., Jaeger, J. et al., Dynamic searchable encryption in very-large databases: data structures and implementation. NDSS 14:23–26, 2014.
20. Nambiar R., Bhardwaj, R., Sethi, A. and Vargheese., A look at challenges and opportunities of big data analytics in healthcare. 2013 IEEE International Conference on Big Data, IEEE. 17–22, 2013.
21. Asri, H., Mousannif, H., Al Moatassime, H. and Noel., Big data in healthcare: Challenges and opportunities. 2015 International Conference on Cloud Technologies and Applications (CloudTech), IEEE: 1–7, 2015.
22. Kepner, J., Gadepally, V., and et al., Computing on masked data: a high performance method for improving big data veracity. 2014 IEEE In High Performance Extreme Computing Conference (HPEC). 1–6., 2014.
23. Dong, X., Li, R. et al., Secure sensitive data sharing on a big data platform. *Tsinghua Sci. Technol.* 20(1):72–80, 2015.

24. Morse, R. E., Nadkarni, P., Schoenfeld, D. A., and Finkelstein, D. M., Web-browser encryption of personal health information. BMC Medical Informatics and Decision Making, 2011.
25. Wicks, P., and Little, M. The virtuous circle of the quantified self: a human computational approach to improved health outcomes. Handbook of human computation, pp. 105–129. Springer New York, 2013.
26. Rahmani, G., Negash, A., and et al., Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. Accepted in Future Generation Computer Systems, 2017.
27. Rao, S., Suma, S. N., and Sunitha, M., Security solutions for big data analytics in healthcare 2015 Second International Conference on Advances Computing and Communication Engineering (ICACCE). 510–514. IEEE, 2015.
28. Hossain, M. S., Muhammad, G. et al., Toward end-to-end biometrics-based security for IoT infrastructure. *IEEE Wireless Commun.* 23(5):44–51, 2016.
29. Hänsel, K., Wilde, N., Haddadi, H., and Alomainy, A.., Challenges with current wearable technology in monitoring health data and providing positive behavioural support. Proceedings of the 5th EAI International Conference on Wireless Mobile Communication and Healthcare. 158–161, 2015.
30. Essa, Y. M., Attiya, G., and El-Sayed, A., New framework for improving big data analysis using Mobile agent. *Int. J. Adv. Comput. Sci. Applic. (IJACSA)* 05(03):25–32, 2014.
31. Zhang, Y., and Liu, D., Improving the efficiency of storing for small files in hdfs. 2012 International Conference on Computer Science & Service System (CSSS). 2239–2242. IEEE, 2012.
32. Aravinth, M. S., Shanmugapriyaa, M. S., Sowmya, M. S., and Arun, An efficient HADOOP frameworks SQOOP and ambari for big data processing. *Int. J. Innov. Res. Sci. Technol.* 1(10):252–255, 2015.
33. Patil, P., Narayankar, P., Narayan, D. G., and Meena, S. M., A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish. Procedia Computer Science 78: 617–624, 2016.
34. K. Gai, M. Qiu, H. Zhao, "Security-aware efficient mass distributed storage approach for cloud systems in big data", IEEE international conference on intelligent data and security (IDS), IEEE, New York, USA. 140–145, 2016.
35. Zhang, Y., Qiu, M., Tsai, C.-W., Hassan, M. M., and Alamri, A., Health-CPS: Healthcare cyber-physical system assisted by cloud and big data. *IEEE Syst. J.* 11(1):88–95, 2017.
36. Essa, Y. M., El-Mahalawy, A., Attiya, G., and El-Sayed, A.., A distributed multiagent architecture for self-healing healthcare data center. 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS 2017), AMA International university Bahrain, 2017.
37. Ali, M., Khan, S., and Vasilakos, A., Security in cloud computing: Opportunities and challenges. *Inf. Sci.* 305:357–383, 2015.
38. Gai, K., Qiu, M., Zhao, H., Xiong, J., Privacy-aware adaptive data encryption strategy of big data in cloud computing", 3rd IEEE International Conference on Cyber Security and Cloud Computing. 273–278, China, 2016.
39. Yu, H., Patnaik, S., Ji, S., Jia, L., and Yang, T., Design and Implementation of Multi-Agent Online Auction Systems in Cloud Computing. *Int. J. Enterprise Inform. Syst. (IJEIS)* 13(1):50–66, 2017.
40. Mohit, P., Amin, R., Karati, A., Biswas, G. P., and Khan, M. K., A standard mutual authentication protocol for cloud computing based health care system. Journal of medical systems, vol.41, no.4, 2017.
41. Park, D., Kang, K., Hong, J., and Cho, Y., An efficient Hadoop data replication method design for heterogeneous clusters. Proceedings of the 31st Annual ACM Symposium on Applied Computing. 2182–2184. ACM, 2016.
42. Islam, M. R., Habiba, N. M., Agent based framework for providing security to data storage in cloud. Proceedings of 15th International Conference on Computer and Information Technology (ICCIT), IEEE, 2012.
43. Islam, M. R., Habiba, M., Kashem, M. I. I., A framework for providing security to Personal Healthcare Records. International Conference on Networking, Systems and Security (NSysS), IEEE, 2017.
44. Yang, W., Wang, S., Hu, J., Zheng, G., Chaudhry, J., Adi, E., and Valli, C., Securing mobile healthcare data: A smart card based cancelable Finger-Vein Bio-Cryptosystem. IEEE Access 6:36939–36947, 2018.
45. Chattaraj, D., Sarma, M., and Samanta, D., Stochastic petri net based modeling for analyzing dependability of big data storage system. Emerging Technologies in Data Mining and Information Security, pp. 473–484. Springer, Singapore, 2019.
46. Chattaraj, D., Sarma, M., Das, A. K., Kumar, N., Rodrigues, J. J. P. C., and Park, Y., HEAP: An Efficient and Fault-Tolerant Authentication and Key Exchange Protocol for Hadoop-Assisted Big Data Platform. IEEE Access 6:75342–75382, 2018.
47. Gai, K., and Qiu, M., Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers. *IEEE Trans. Indust. Inform.* 14(8):3590–3598, 2018.
48. Hospital Compare datasets, Centers for Medicare & Medicaid Services, https://data.medicare.gov/data/hospital-compare, 2018.
49. National Toxicology Program, Chemical Effects in Biological Systems (CEBS) database, 2018.
50. Li, W., Yang, Y., and Dong Y. ,A novel cost-effective dynamic data replication strategy for reliability in cloud data centres. 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing. 496–502. IEEE, 2011.