International Journal of Cloud Applications and Computing
Volume 8 • Issue 2 • April-June 2018

An Affordable Hybrid Cloud Based Cluster for Secure Health Informatics Research

Basit Qureshi, Prince Sultan University, Riyadh, Saudi Arabia

ABSTRACT

This article describes how a major risk factor in the deployment of patient health records systems in the cloud is the security and privacy of data. Hybrid cloud solutions have been proposed that leverage the public and private cloud deployment to manage and alleviate accessibility, access control and privacy concerns. This article presents a privacy preserving and secure architecture for data acquisition, storage, processing and sharing. The proposed architecture is composed of a public cloud-based services that interact with a low-cost cloud computing cluster (LoC4) as a backend. A lightweight data security eco-system based on attribute based encryption is developed to provide security for public cloud-based data storage. Performance of the deployment is evaluated in a real-time deployment environment. The results show that the proposed ABE-based system is 2.3 times faster than AES-based for a variety of sizes of data blocks. It is further noted that the low-cost and affordability of LoC4 platform offers excellent opportunities for academic research in cloud based health informatics.

KEYWORDS

Cloud Computing, Low Cost Cluster, Personal Health Records, Single Board Computers

1. INTRODUCTION

Electronic health record (EHR) and personal health record (PHR) systems are widely available and use different technologies and standards (Bahga, 2015; Sun, 2013). The variety and size of medical health records data makes it difficult for researchers to accurately and easily integrate data from various sources. Due to the high cost of building and maintaining specialized data centers many healthcare providers have been outsourcing PHR systems to third part cloud service providers such as Microsoft Health Vault, Google Health, General Electric's Centricity Patient Online, eClinicalWorks and many more. Cloud computing provides storage for large scale data on external servers allowing researchers and developers to easily access the data (Fang, 2016). Although the technology is being used in recent years with many benefits such as reducing the cost for maintaining servers whereas improving the availability of the systems, there have been wide spread privacy concerns since personal healthcare information can be exposed to unauthorized parties.

Due to the sensitivity of personal information, the cloud based PHRs could potentially become targets of various malicious behaviors. Furthermore, Guo et. al. in (Guo, 2016) and (Abbas, 2017) describe various security issues and threats in cloud computing environment. In (Asija R., 2016), researchers develop a security aware public cloud based SaaS model for healthcare applications embedding XML based meta data for PHRs. Works presented in (ZhouJ, 2015), (ZhouCao, 2015) and (Li, 2013) have proposed several cloud-based secure systems, especially cloud-based PHR systems using encryption where PHR data is encrypted and stored in cloud based storage. In (Man, 2017),

DOI: 10.4018/IJCAC.2018040102

(Wang, 2014) and (Zhou, 2015) authors deduce that any public cloud based security model may have inherent threats of trust, security and privacy. They highlight that the complexity of key management in encrypted PHR systems increases for the owners and users resulting in Quality of Service issues. (Liu, 2017) note that hybrid cloud environment integrating the public and private cloud infrastructure is a more applicable solution for PHR sharing.

In this paper, we take inspiration from (Liu, 2017) to investigate requirements for the deployment of a secure hybrid cloud environment for a PHR system. A two-tier hybrid cloud architecture is proposed addressing security and privacy preserving mechanisms for information storage and retrieval in the cloud. A light weight data security eco system is developed to provide Attribute Based Encryption (ABE) for public cloud based data storage that enables multiple users to share data. We further introduce LoC4, an affordable and low-cost cloud computing cluster for health informatics research in universities and academic institutions. The proposed hybrid cloud based PHR system is developed and deployed using an experimental setup coupling public and private cloud based implementations. We conduct several experiments to analyze the performance criteria including computation efficiency, storage and network efficiency, task completion rate, response time etc. Furthermore, we compare the performance the proposed Encryption scheme and compare the response time for task execution for various sizes of data blocks. Results show that LoC4 provides an affordable hybrid cloud based backend solution for low cost, energy efficient and secure deployment.

The contributions of this paper are in the following:

- Requirements of a secure hybrid cloud environment for a PHR system are presented. Based on an extensive investigation, a three-layered architecture is proposed including: i) secure data acquisition, ii) storage of PHRs in public cloud and iii) encryption and privacy layers;
- A two-tier, light weight encryption eco-system is developed encompassing public cloud service and private cloud based cluster deployment;
- Design details and implementation of a LoC4 cluster using SBCs is provided. The Cluster deployment parameters are optimized for performance and energy efficiency;
- A light weight middleware for communication and interaction between public cloud service and the LoC4 Cluster is developed. This middleware enables real time performance measurement of the proposed system.

The remainder of this paper is organized as follows: Section 2 provides description of requirement specifications for hybrid cloud architecture for PHR systems. Section 3 details the two-tier light weight eco-system. Section 4 provides design details for the LoC4 cluster as well as performance specific variables for deployment of Apache Hadoop on the cluster. Performance evaluation of the proposed system is presented in section 5. Section 6 presents related works with conclusions and future works presented in section 7.

2. REQUIREMENTS SPECIFICATIONS FOR A CLOUD BASED PHR SYSTEM

A hybrid cloud based PHR system is composed of various subsystems interacting in a public and private cloud environment. Figure 1 shows various parts of a hybrid cloud deployment of a PHR systems for a Healthcare provider (HCP). Data deemed secure and private is kept in a data center composed or various clusters whereas unsecure data or data with no privacy concerns is kept for public access in a public cloud. In this section, we provide requirements specifications for a secure hybrid cloud based PHR system.

2.1. Storage in Cloud Based PHR System

A cloud based PHR system needs to address secure data acquisition and storage requirements before further processing of data. Figure 2 shows an architectural map of the three layers of data flow in a PHR system. In what follows, we describe requirements for these data flows.

At the acquisition stage, data is collected using various mechanisms for data entry. It could be as complex as a Wireless Body Area Network (WBAN) (Page, 2014) (Page, 2015) consisting of various wireless wearable sensors for specific medical applications such as blood pressure, thermometer etc. These sensors and devices can connect to a data collection device using Bluetooth, Zigbee protocols or similar wireless communication medium, and transmit the information to an intermediate cloudlet for synchronizing and processing information (Soyata, 2016). This data could be acquired, stored on the cloudlet or transmitted on demand for access by the HCPs.

Reliable, scalable and secure storage in the cloud is perhaps the most important functionality of Cloud based PHR systems. Storage services based on public clouds such as Microsoft's Azure storage service and Amazon's S3 provide customers with scalable and dynamic storage. These services allow users to move their data into the cloud and avoid costs of building and maintaining a private storage infrastructure using pay as you go cost model. This offers various benefits such as availability (i.e., being able to access data from anywhere) and reliability (i.e., not having to worry about backups) at a relatively low cost. Consumers can trade privacy for convenience of software services, however enterprises and government agencies are more concerned with the preservation of confidentiality and integrity of data.

2.2. Data Privacy Requirements

The Health Insurance Portability and Accountability Act (HIPAA) (HIPAA US, 2006) states that data privacy must be protected within every layer of a PHR system. Here we describe the requirements for each layer of the system:

Figure 1. A hybrid cloud environment for a PHR system at HCP

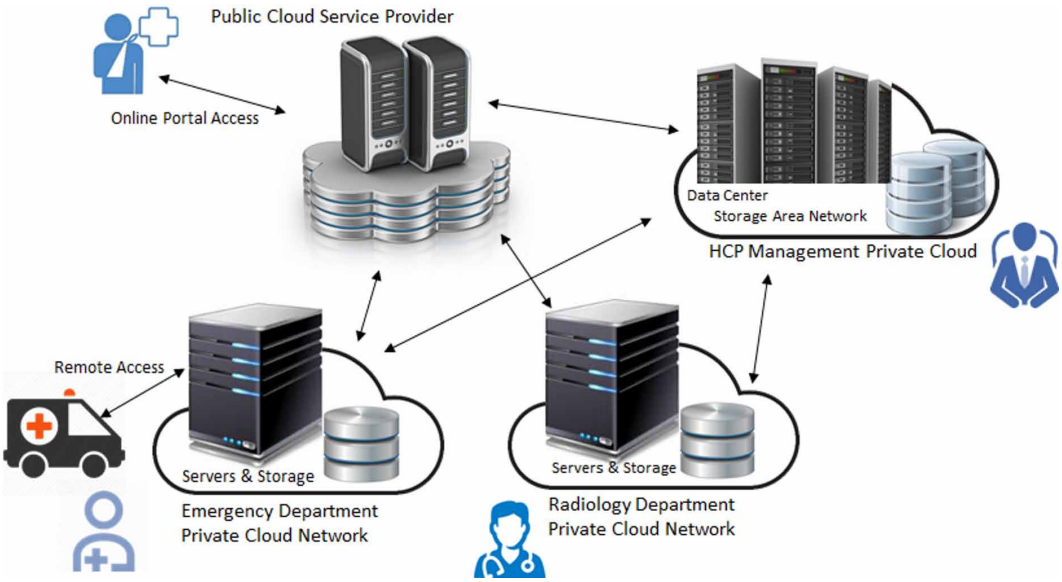
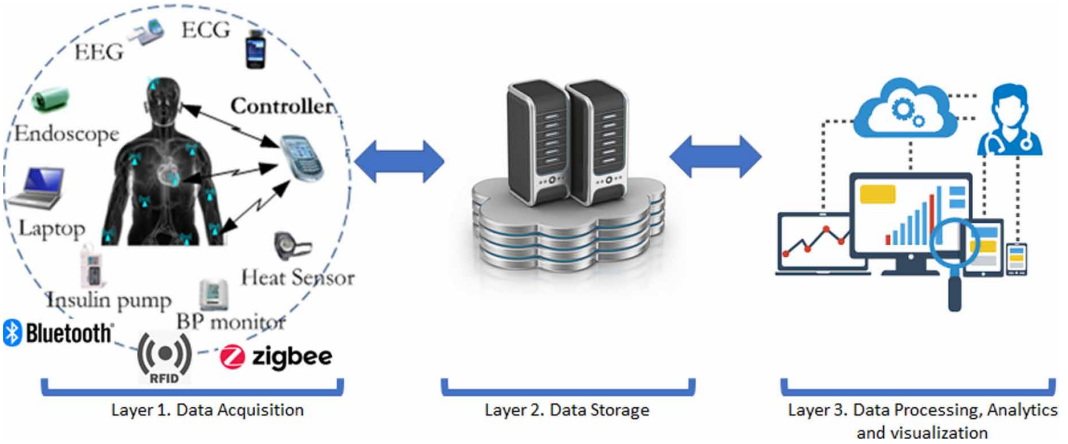


Figure 2. Architectural layout of cloud based PHR system requirements



1. **Data acquisition Layer:** The acquisition layer in Figure 2 is composed of Wireless Body Area Network sensor devices. Typically, these devices have limited computational capacities and operate on finite battery power. Encryption schemes used to protect the communication within BAN sensors and BAN-to cloudlet communications should not be computationally intensive. (Page A., 2014) present a privacy preserving mechanism for secure information exchange using Advanced Encryption Standard (AES);
2. **Data Storage Layer:** There can be various stakeholders involved in storage, retrieval and sharing of data from the system including, patient, doctor/nurse, emergency unit staff etc. Allowing Cloud service providers to manage sensitive information such as EHR could potentially raise security issues. An untrustworthy Cloud service provider may leak PHR to medical companies for profit. It is suggested to encrypt all the data placed in the Cloud, however, this could have a severely negative affect on the Quality of Service as the volume of data grows. Furthermore, the cost of encryption also increases linearly in time and space.

Another issue is the QoS in searching and indexing of data. To enable searching over the data, a user must either store an index locally, or download all the (encrypted) data, decrypt it and search locally. Both approaches negate the benefit of cloud computing and are therefore poor choice for processing of large scale data thus increasing the overhead of security and volume of communications.

Conventional encryption techniques such as symmetric key encryption does not work as it requires both parties to use a symmetric key. Asymmetric encryption techniques such as Identity based encryption (IBE) may work, but it requires one party to decrypt, other than this party, no one can decrypt the data. It can be suitable in a scenario when a data record is shared only between two parties, however for multiple actors such as doctors, nurses, insurance companies, patients, etc., management becomes an overhead:

3. **Data Processing, Analytics and Visualization Layer:** To process data, it must first be decrypted. Decryption of this data in the intermediate stage requires trusted storage infrastructure possibly at the site of the HCP necessitating a private cloud infrastructure. After the data has been processed, it needs to be removed from the intermediate storage. Furthermore, this encryption and decryption of data adds to overall overhead of computation, processing and visualization of data. Efficiency, reliability and integrity must be preserved for data processed in the system.

It is imperative that in an enterprise environment, many stakeholders may have access to various kinds of data. Doctors, patients, nurses, emergency unit staff, medical insurance personnel etc. to name a few, require strict access control mechanisms in place. Furthermore, in WBAN environment, various devices could be owned by a consumer or even shared by others. For any efficient solution, it is crucial to address the above-mentioned issues thoroughly and effectively.

3. SECURE HYBRID CLOUD ARCHITECTURE FOR PHR SYSTEMS

In this section, we present the design and architecture for a secure hybrid cloud based PHR system. The proposed architecture considers the requirements of data acquisition, storage and analysis while preserving the data privacy requirements. We present a high-level architecture of the proposed system in Figure 3 which incorporates various components into public and private cloud tiers. All services interact together to provide secure, reliable and scalable solution.

3.1. Public Cloud Service

The public cloud service consists of various components including encrypted data storage, open access data storage, Health analytics engine, Audit Logs, Enterprise Access control list and Key verification module for data integrity. We adopt attribute-based encryption (ABE) as the cryptographic primitive to serve this purpose. By using ABE, access control definitions express the user attributes (meta-data) or data (PHR) which enable a user to selectively share their data without knowledge of the other party's identity.

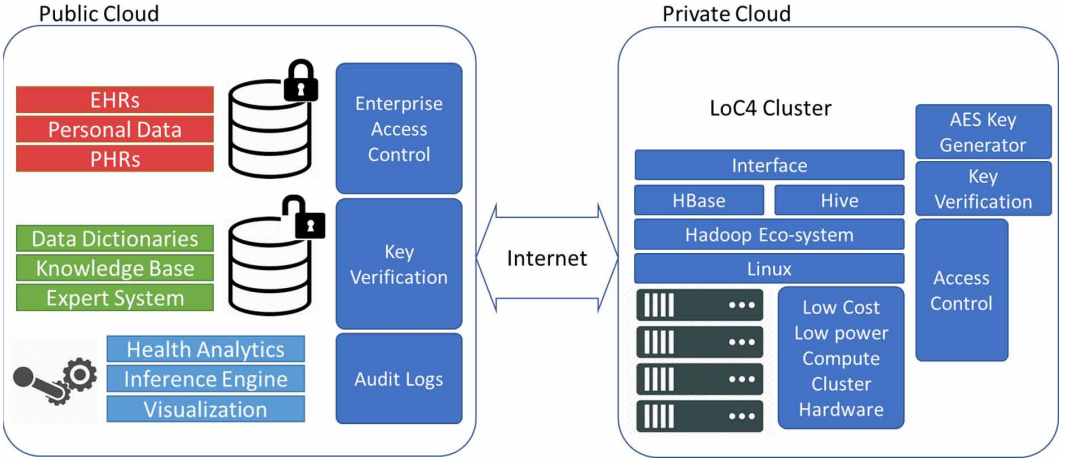
Privacy critical data such as personal health records need to be secured on the public cloud, whereas data with low privacy requirements such as data dictionaries etc. or intermediary data such as data generated from health analytics or visualization, many not require stringent security measures. It is important to classify these kinds of data to improve the efficiency of the system by leveraging the amount of computations and time consumed in communication. Large volumes of data generated at the Data acquisition layer needs to be stored and shared with appropriate consumers with valid credentials. Access Control parameters are defined to identify appropriate levels of users mapped with data in storage. This mechanism allows stakeholders with appropriate credentials to access and share data. Access Control parameters are defined in the private cloud service and are periodically updated in the public service.

We develop a custom encryption approach that is efficient and reliable and is described further in this section. Audit logs are stored for all transactions carried out by various consumers. These logs are beneficial for forensics in determining illegal behavior. The public cloud service is implemented in an open access system and deployed on a public cloud service provider.

3.2. Private Cloud Service

The private cloud service essentially hosts a cluster of machines implementing a small data center at the premises of the enterprise. The infrastructure is secured in the enterprise locale with physical, data and network security measures in place. The data center hosts low cost, efficient and eco-friendly LoC4 cluster. The machines in the LoC4 cluster run open source operating system such as Linux with Hadoop eco-system deployment for distributed task processing and management. Access control information for the enterprise is stored in HBase and Hive data warehouse which are encrypted using commonly available encryption techniques. Additional modules for encryption key generation and verification are implemented in the private cloud. These are necessary for efficient and reliable storage and sharing of data. Communication takes place through the interface which implements a light weight communication protocol for data transmission between the public and private cloud services. Section 4 details the design and implementation of the LoC4 cluster which is deployed using low cost low power Single Board Computers (SBCs).

Figure 3. High-level architecture



3.3. Security and Privacy

Here we address the security implications and proposed solutions at the various layers of data in a PHR system.

3.3.1. Data Acquisition Layer

Data acquisition devices usually have limited capabilities for storage or processing large scales of data. This implies that any encryption schemes used for security in data acquisition must not be computationally intensive. An obvious solution is to use the Zigbee protocol which is based on the AES encryption scheme and can easily be implemented. Communicating devices must agree on a secret-key before using AES encryption by using generic key exchange algorithm such as Diffie-Hellman (DH) (Diffie, 2006). Irrespective of the kind of encryption scheme, a consumer must agree on key(s) to encrypt/decrypt messages with a service provider. Advanced Encryption Standard (AES) is one of the most widely used symmetric key encryption algorithms and is accepted as an industry and a government applications standard. AES is optimized for speed, low memory footprint and energy efficiency. Its low resource intensity allows AES to run on a wide range of hardware platforms. Communication of devices can be also secured by using biomedical signals as noticed in (Venkatasubramanian, 2010).

3.3.2. Secure Data Storage, Sharing and Processing

We use a variant of Attribute based encryption (ABE) scheme. Attributes set is defined where each member of this set is given associated private keys which are generated by third parties. The HCP provides encryption with a token, any user with appropriate attribute set can decrypt the data as long as it fulfils the token. The token can be generated such that the retrieval procedure reveals nothing about the files or the keywords (meta-data) except that the files contain a keyword in common. This light weight technique addresses the inefficiency of downloading an entire encrypted block and processing only a limited part of it. Furthermore, the coupling of access control mechanisms (attributes) allows reliable sharing of appropriate data with proper stakeholders.

The data is encrypted using an access control policy based on credentials. Only the user whose credentials satisfy the access policy can have access to the data. The attributes can be the profession (e.g., Doctor, Nurse) or the department (e.g., Cardiology, Emergency) of the user. An access policy can be defined as joints and disjoints with various attributes such as:

(Doctor AND Pediatrics) OR (Nurse OR Intensive-Care-Unit)

Which gives access to a Doctor in Pediatrics Department or a Nurse or an Intensive-Care-Unit staff member.

We present a publisher/subscriber model for the encryption and decryption of Data Blocks. Two algorithms are defined PushDataBlock (Algorithm 1) and PullDataBlock (Algorithm 2), that control read / write of data blocks using a combination of key and access control meta data, in the public cloud service. In the following text, we describe the working of these two algorithms to enable secure data communication and storage of a data block in the public cloud service.

Algorithm 1. PushDataBlock

```
input: DataBlock B_i, SecretKey S_k, AccessControlMetaData AC_MDi
output: Encrypted DataBlock EB_i
  for i = 1 to n - 1 do
     KeyGenerator(B_i, S_k, AC_MDi)
     PushCloud(EB_i)
  end
```

Algorithm 2. PullDataBlock

```
input: SecretKey S_k, Keyword
output: Decrypted DataBlock DB_i
  for i = 1 to n - 1 do
     RequestToken(AC_MD, S_k, Keyword)
     AcquireToken(S_k)
     PushToken(token, S_k)
     PullDataBock(DB_i)
     VerifyDataBlock(DB_i)
 end
```

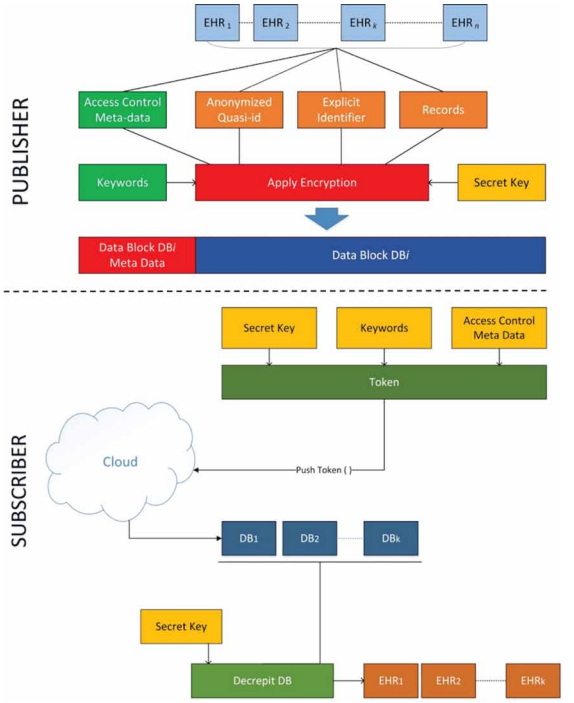
Each subscriber is allocated a Key S_k to facilitate access to the private cloud service, these keys are managed by the administrators at the enterprise and assigned to stakeholders with appropriate access control credentials.

Assuming a data block B_i was generated (data acquired from WBAN cloudlet, etc.) consisting of one or many records / files and is ready to be uploaded to the public cloud storage, the publisher would call the *PushDataBlock* with an access control meta data AC_{MDi} for this data block B_i.

The *KeyGenerator* is responsible for properly indexing and encrypting the data block. KG is responsible for encrypting various data blocks using access control information provided and appends metadata for identification of datablocks such as timestamp, size, keywords, ownership etc. It encrypts the data block in such a way that given a *token*, a subscriber can retrieve pointers to the encrypted records/files that contain a *keyword*. Without appropriate *token*, the data block would not be decrypted. The tokens cannot be generated without a secret key S_k. It must be noted that the retrieval procedure does not reveal any details within the data block except only the searchable keywords. Once the *KeyGenerator* has encrypted the DataBlock B_i, *PushCloud*(B_i) is called to push the data block to the public cloud storage.

When a subscriber intends to access a data block for processing or sharing, it needs to call the *PullDataBlock* with the secret key S_k, and the access control meta data AC_{MD}. *RequestToken* call is made for a search keyword term to the private cloud service. Based on the access control policy, the *AcquireToken*(S_k, AC_{MD}) generates a token which is then sent to the public cloud service using

Figure 4. Publisher subscriber model for encryption of data blocks in the framework



the *PushToken*(*token*). The public cloud service uses the token to find the appropriate encrypted documents. The *PullDataBlock* call returns the selected documents as data blocks to the subscriber. At any point and time, the subscriber can verify the integrity of the data blocks by calling the *VerifyDataBlock* call.

Figure 4, shows the publisher subscriber model used for encryption of Data Blocks. In the next sections, we present details for deployment of the proposed model on a LoC4 cluster.

4. DESIGN OF LOC4 CLUSTER

This section presents the architecture and configuration of the LoC4 cluster deployed in this experimental study.

4.1. Single Board Computers

A single-board computer (SBC) is a complete computer built on a single circuit board (Baun, 2016). A SBC incorporates microprocessor(s), memory, I/O as well as host of other features required by a functional computer. A wide range of programming languages are supported by these platforms. In this work, we investigate use of Raspberry Pi Model 3B (RPi, 2017) as well as Hard Kernel Odroid Xu-4 (Odroid, 2017). A summary of various features of these computers can be seen in Table 1.

4.2. LoC4 Cluster Deployment

This section presents the architecture and configuration of the LoC4 cluster deployed in this experimental study.

Table 1. Comparison of Raspberry Pi 3B and Hard Kernel Odroid computers

	Raspberry Pi Model 3B	Odroid XU-4
Processor (CPU)	1.2 GHz Quad core 64-bit ARM Cortex-A53	Samsung Exynos5 Octa ARM Cortex-A15 (@ 2.0 GHz) and Cortex-A7 (@1.3GHz) CPUs
GPU	400MHz Video Core IV multimedia	Mali T628 Open GL 3.0
Onboard RAM	1 GB LPDDR2-900 SDRAM (at 900MHz)	2GB LPDDR3 at 933MHz
Ethernet / Network	10/100 MB Ethernet RJ45 Jack. Onboard 2.4GHz Wi-Fi 802.11n and Bluetooth 4.1	10/100/1000 MB Ethernet RJ45 Jack
Storage	Micro SD Card	Micro SD Card and eMMC 5.0 flash storage
Audio/Video	Micro HDMI	HDMI (standard) supports 1080p video
Power Consumption	2.0 W (idle) 3.8 W (under load)	2.5 W (idle) 4.5 W (under load)
USB Ports	4x USB 2.0 ports	1x USB 2.0, 2x USB 3.0
Released	2016	2015
Price (US$)	35 $	79 $

4.2.1. Design of the Cluster

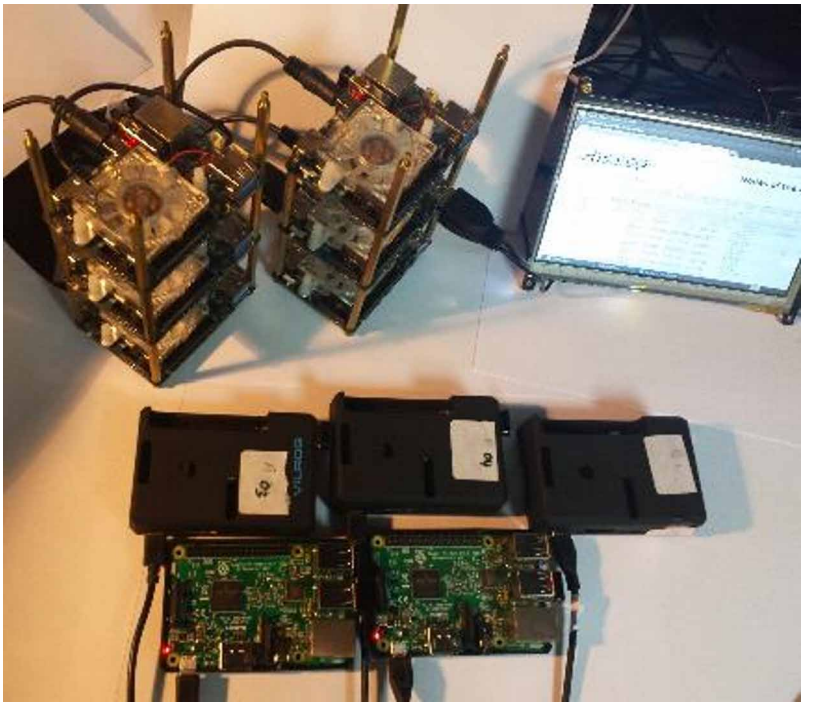
The LoC4 cluster is composed of 11 SBCs of which five are RPi Model 3B computers in addition to six Odroid Xu-4 computers interconnected with power supplies, network cables, storage modules, connectors and cases. All the Raspberry Pi computers are equipped with 16 GB Class-10 SD cards for primary bootable storage. The Odroid Xu-4 devices are equipped with 32GB eMMCv5.0 modules. The Odroid Xu-4s are housed in a compact layout racks using M2/M3 spacers, nuts and screws. The Raspberry Pi 3B's are housed in cases and connected to the rest of the cluster. Currently each Raspberry Pi computer is individually supplied by 2.5Amp power supply; each Odroid Xu-4 computer is supplied by a 4.0Amp power supply that provides ample power for running each node. Each SBC's network interface relates to a Cat6e Ethernet cable through the RJ-45 Ethernet connector. All Ethernet cables connect to a 16-port Cisco switch which in turn is connected to the university network equipment. The size of the cluster can be easily scaled by introducing a core switch that connects to the 16-port switch. Additional nodes can be included in the network configuration scaling the size of the cluster.

For Hadoop deployment, one of the Xu-4 SBC is used as the master node whereas the rest of the nodes server as slaves' nodes. We compare the performance of Xu-4 sub cluster (composed of five Xu-4 nodes) with RPi sub cluster also composed of five RPi 3B nodes. Figure 5 shows the layout of the LoC4 cluster. The purchase cost for all equipment for the LoC4 Cluster was $1179.

4.2.2. Hadoop Deployment

Apache Hadoop (Hadoop, 2014) is an open source framework that provides distributed processing of large amounts of data in a datacenter. Hadoop version 2.6.2 was installed due to availability of YARN daemon which improves the performance of the map-reduce jobs in the cluster. To optimize the performance of these Clusters, yarn-site.xml and Mapred-site.xml were configured with 512 MB of resource size allocation. The primary reason for this setting is the limited amount of RAM available in the RPi Model 3B. The Raspbian operating system as well as the shared GPU memory bus consume over 200MB or RAM out of a total of 1GB. The default container size on the Hadoop Distributed File System (HDFS) is 128 MB. Each SBC node was assigned a static IPv4 address based on the configuration and all slave nodes were registered in the Master node. Table 2 provides detail of important configuration properties for the Hadoop environment.

Figure 5. The LoC4 cluster with 6 Xu4 (rack mounted) and 5 RPi 3B in cases



For YARN Resource manager, we allocated up to 4 cores which means up to 4 containers can execute per node (one container per core). The replication factor for HDFS is 2 which means only two copies of each block would be kept on the file system. The LoC4 Cluster was tested extensively for performance using TeraSort benchmark.

The LoC4 cluster was connected to the Internet using a static IP address for experimental evaluation presented in the next sections.

5. PERFORMANCE EVALUATION

In this section, we present a performance evaluation study of the proposed Hybrid Cloud based PHR system. We first look at the performance evaluation of the LoC4 cluster in terms of processing speed, storage read/write and networking. The objective of this study is to highlight the intrinsic capabilities of SBCs as well as to showcase the performance of these SBC when deployed as a cluster. We use popular benchmarking techniques widely used in the literature to highlight the capabilities of LoC4. Second, we present performance in terms of Response time for the *PushDataBlock* and *PullDataBlock* algorithms presented in section 3.

5.1. Evaluation of LoC4

5.1.1. CPU Execution Times

The benchmark suite Sysbench[1] was used to measure the CPU performance. We execute the Sysbench benchmark[2], testing each number up to value 10,000, if it is a prime number, for *n* number of threads. Since each computer has a quad core processor we run the sysbench CPU test for 1, 2, 4, 8 and 16 threads. We measure the performance for this benchmark test for Raspberry Pi Model 2B, Odroid Xu-4 as well as an Intel i7 3.0 GHz 4th Generation personal computer (for comparison purposes).

Table 2. Hadoop configuration properties for LoC4 cluster

mapred-site.xml Properties	Value
yarn.app.mapreduce.am.resource.mb	512
mapreduce.map.cpu.vcores	1
mapreduce.reduce.cpu.vcores	1
mapreduce.map.memory.mb	512
mapreduce.reduce.memory.mb	512
mapreduce.input.fileinputformat.split.minsize	8 MB
YARN-site.xml Properties	
yarn.nodemanager.resource.memory-mb	1024
yarn.nodemanager.resource.cpu-vcores	1
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	1024
yarn.scheduler.minimum-allocation-vcores	1
yarn.scheduler.maximum-allocation-vcores	4
yarn.nodemanager.vmem-pmem-ratio	2
hdfs-site.xml Properties	
dfs.replication	2

The CPU execution times scale well with increased number of threads. Sysbench test runs with n = 2 and n = 4 threads significantly improves the execution times performance for all processors by 50%. With n = 8 and n = 16 threads the test results yields almost similar execution times with little improvement in performance. Table 3 shows results for the Sysbench. The execution times for Odroid Xu-4 are 6 times better as compared to Raspberry Pi Model 3B. The increased number of threads (larger than 8) does not provide gain in performance of Odroid Xu-4 over Raspberry Pi. It is also noted that the execution time for Raspberry Pi is further extended with larger *n*.

5.1.2. Storage Performance

Poor storage read/write performance can be a bottleneck in clusters. Compared to a server machines, an SBC is handicapped in terms of availability of limited storage options. In our experimentation, the Raspberry Pi's were equipped with 16GB SanDisk Class 10 SD Cards, whereas the XU-4 devices were equipped with 32GB eMMC memory cards. Both memory cards were loaded with bootable Linux distributions.

FIO[3] was used for benchmarking of random read and write with various block sizes. We consider the throughput with 8 threads each working with a file of size 512MB with a total 4GB of data. These parameters were set specifically to avoid buffering and caching in RAM issues which is managed by the underlying operating systems that can distort the results; i.e. the data size (4GB) selected is larger than the onboard RAM available on these devices. Table 4 shows the comparison of buffered and non-buffered random read and write from both devices with block size 8KB. As can be seen from Table 4, the read throughput (buffered) of Odroid with eMMC memory is at least twice as fast as the Class 10 SD Card on the Raspberry Pi whereas the non-buffered read is more than three times better. Similarly, for buffered write operations, Odroid Xu-4 with eMMC module throughput is more than twice better when compared to the Class 10 SD Card in Raspberry Pi.

Table 3. The average CPU execution time for nodes with n threads

	CPU Cores	Clock Rate GHz	CPU Execution Time with *n* threads 1	2	4	8	16
Raspberry Pi 3B	4	1.2	382.2	192.07	96.31	98.69	99.73
Odroid Xu-4	8	2.0 and 1.4	83.38	41.68	25.336	17.66	18.02
Intel i7 4th Gen	4	3.0	8.508	4.272	2.228	2.27	2.23

5.1.3. Network Performance

was measured using the iperf 4v3.13 with the NetPIPE5 benchmark version 3.7.2. Through various sets of runs, iperf states the network throughput to be 82-88Mbits per second for the RPi as well as the Xu-4 SBCs. The NPtcp, NetPIPE benchmark using TCP protocol, involves running transmitter and receiver on two nodes in the cluster. In our experimentation, we executed the receiver on the LoC4 master node (Xu-4 device) with 1000 KB as maximum transmission buffer size for a period of 240 milliseconds. The transmitter was executed on the individual SBCs one by one. As can be seen from Figure 6, the network latency for both devices with small payload is almost similar. As the payload increases we observe slight increase in network latency between both sets of devices. On the other hand, we observe a spike in throughput with message size 1000 bytes. This proves that network layer communication overhead dominates the overall transfer time for small payloads. For larger payloads, the communication rate correlates with data rate at the network link, utilization of the communication medium at the time or the traffic at the network switch. Contrasting the performance of Xu-4 and RPi SBCs we note the visible difference in throughput between the two, which is due to the poor overall Ethernet performance of the Raspberry Pi.

5.1.4. Hadoop TeraSort Benchmark

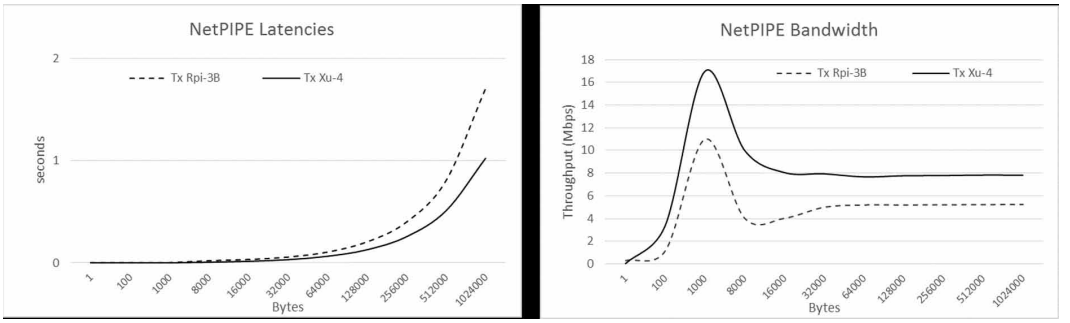
Hadoop provides various benchmarks for performance of the Hadoop cluster. TeraSort is used in this study to observe the performance of LoC4 cluster in terms of task execution time. TeraSort combines testing the HDFS and MapReduce layers of a Hadoop cluster and consists of three MapReduce programs, TeraGen, TeraSort and TeraValidate. TeraGen is used to generate large amounts of data blocks, consequently, TeraGen is a write intensive I/O benchmark. TeraSort generates set of sample keys by sampling the input data generated by TeraGen before the job is submitted and writes the list of keys into HDFS. The input and output format, which are used by all three MapReduce programs, reads and writes the text files in the correct format. The TeraSort benchmark is CPU bound during the map phase and I/O bound during the reduce phase. TeraValidate verifies that the output of the TeraSort is globally sorted. We run TeraGen, and TeraSort on LoC4 cluster with 400MB, 800MB, 1,600MB and 3,200MB data sizes.

We observe the job execution time for each run for comparison and analyze the performance on the cluster. Since TeraGen is I/O intensive, the write speeds of the memories/storage in corresponding

Table 4. Read and write throughput (KB/sec) for individual devices in the clusters using FIO

	Read Throughput (KB/second)		Write Throughput (KB/second)	
	Buffered	**Non-Buffered**	**Buffered**	**Non-Buffered**
Raspberry Pi 3B with 16 GB Class 10 SanDisk SD Card	3,688	2,341	1,318	1,267
Odroid Xu-4 with 32GB eMMCv5.0 Module	9,197	6,883	3,220	2,519

Figure 6. NetPIPE benchmark results for Xu4 and RPi devices considering latencies and bandwidth with data size in terms of bytes on the x-axis



nodes in the clusters plays a major role in degrading the overall job completion time. The average job completion times for RPi and Xu-4 sub-clusters can be seen in Figure 7. The completion time for TeraGen on Xu-4 and RPi is correlating for various data sizes with Xu-4 being 3 times faster compared to RPi. We compute the execution time for TeraSort. For all experiments, we use the same number of map and reduce tasks on each cluster. The TeraSort benchmark is CPU bound during the map phase, i.e. reading input files and sorting tasks are carried out whereas it is I/O bound during the reduce phase, i.e. writing output files in the HDFS. We observed that 33-40% of job completion time occurred in map phase while 49% or more time was spending in reduce tasks overall for the majority of TeraSort jobs run on the cluster. We observe that as the data size increases the execution time also increases for RPi. Figure 7 also shows the comparison of the ratio of execution time degradations for RPi against Xu-4 sub-cluster. For smaller data size of 400MB the Xu-4 is 3 times faster whereas for larger data size of 3.2 GB, the Xu-4 is 3.4 times faster.

5.2. Evaluation of Hybrid Cloud Based PHR

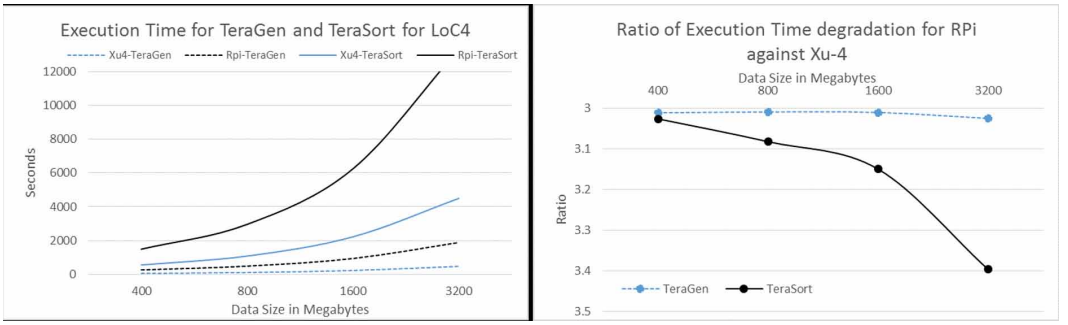
In this section, we evaluate the performance of the Hybrid Cloud based PHR system presented in section 3, in two phases. First, we write a Linux based script to encrypt data blocks of various sizes using AES and push these to a public cloud service and observe response time. Next, we compare the response time with the ABE based *PushDataBlock* encryption algorithm to determine the cost of encryption in terms of response time. Second, we measure response time at the public cloud service by executing the subscribe modules. The data blocks are pulled from the cloud based on user provided attributes. For this experimental study, xml files of various sizes are used to mimic datablocks. Each file is assigned with access control information and useful meta-data such as filename, keywords, etc.

5.2.1. Response Time for Publisher

The purpose of this set of experiments is to validate the functioning of all modules for publisher access in the Hybrid Cloud based PHR System. A Ubuntu based virtual machine instance with 4 CPU cores, 2GB RAM and 50GB storage is launched in DreamHost. A standalone Hadoop 2.6.2 installation is initiated on this virtual machine. Various sets of xml files are generated with different sizes and are pushed into a HDFS directory from the master node in the LoC4 cluster. These files are encrypted using Advanced Encryption Standard (AES) with 128-bit length keys. We measure the response time in terms of task completion time, at the master node in the LoC4 cluster.

Next, we write a light weight middleware using bash script which is responsible for executing the publisher components of the system. The *KeyGenerator* Module is called to encrypt xml files of various sizes (1.8 MB to 394MB) using the access control parameters including up to 40 attributes and meta data for files. Access control parameters were generated for four users (Doctor, Nurse, Manager and Patient). These files were then pushed to the public cloud using the *PushDataBlock*. Again, we

Figure 7. TeraGen and TeraSort execution time for LoC4 cluster



measure the response time in terms of task completion time at the LoC4 cluster master node. Table 5 shows the comparison of response time (time to completion) of both sets of experiments. Results show that the proposed scheme using ABE is on average 2.3 times faster than the AES based scheme in terms of response time for task completion. Figure 8 shows the response time for execution of the AES based technique versus the proposed mechanism. It must be noted however, as the Data block size increases the task completion time for both also increase linearly.

5.2.2. Response Time for Subscriber

These experiments validate the subscriber modules of the proposed system, namely, *KeyGenerator*, *PullDataBlock*, *RequestToken* and *AcquireToken*. Bash script is written in the DreamHost based virtual machine requesting data blocks based on a query. Given the access control parameters and the search keywords as part of the query, *RequestToken* sends a request to the LoC4 Cluster. The LoC4 *AcquireToken* provides a token which is sent to the public cloud service. Using this token, *PullDataBlock* decrypts the block that can be accessed by the user on the public cloud.

Our results show that the subscriber is successfully implemented with reasonable response times. It must be noted due to the settings in the SBC based LoC4 cluster, the minimum response time for

Figure 8. Response time for task execution of data blocks using AES and proposed schemes

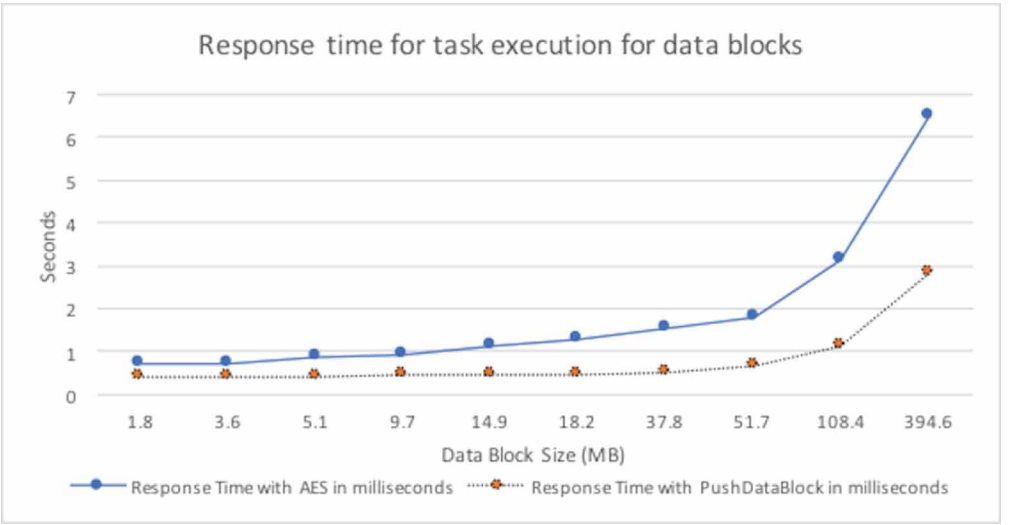


Table 5. Response time for various publisher access tasks

DataBlock(s) Size in MB	Response Time With AES in Milliseconds	Number of Attributes	Response Time With *PushDataBlock* in Milliseconds
1.8	7098.2	20	4009.9
3.6	7211.9	20	4101.2
5.1	8633.6	20	4208.1
9.7	9221.1	20	4304.8
14.9	11013.9	20	4408.0
18.2	13009.8	40	4610.0
37.8	15291.3	40	5022.6
51.7	18050.2	40	6814.0
108.4	31167.0	40	11089.1
394.6	64985.2	40	28223.1

executing Hadoop jobs is approximately 3 seconds. Given these conditions the response time for smaller workloads i.e. less than 50MB for the proposed system is within two seconds. It is observed, as the size of the *DataBlocks* increases, the time to completion also increases linearly.

6. RELATED WORKS

In recent times, many cloud service providers such as Google, Microsoft and Amazon have provided PHR services through the cloud. A PHR inherently saves privacy data of individuals which leads to the need for prevention of this data from unauthorized access. Gou et.al in (Gou, 2016) present a survey on various security issues and challenges in cloud computing environment.

In (Asija, 2016), researchers develop a security aware public cloud based SaaS model for healthcare applications embedding XML based meta data for PHRs. Work detailed in (ZhouCao, 2015) presents a privacy preserving protocol for cloud assisted e-healthcare data sharing. Authors in (Li, 2013) proposed a cloud-based PHR systems framework using encryption where PHR data is encrypted and stored in cloud based storage. (Qureshi, 2014) provide a cloud based framework for big data applications in healthcare. The proposed framework leverages the benefits of cloud computing for addressing many issues in traditional PHR systems. A proof of concept is provided although no details of implementation or deployment of the framework are available. Basu et. al in (Basu, 2012) present Fusion, an experimental open, cloud-based platform for large-scale, low-cost delivery of healthcare applications. Fusion addressed use of patient-centric management of electronic health records and facilitates the secure and seamless sharing of EHRs among stakeholders within a healthcare system. (Peyton, 2017) provide a Health informatics system design approach to support community based care delivery.

Authors in (Wan, 2013) present a detailed study on cloud-enabled architecture and its applications in pervasive healthcare systems. They provide a prototype implementation of the proposed mobile healthcare system. Wang et.al. in (Wang, 2014) use a case study of mobile-cloud based electrocardiograph monitoring and analysis and develop a hybrid mobile-cloud prototype. The experimental results show that the proposed approach can enhance the conventional mobile-based medical monitoring. (Alghabban, 2017) develop a cloud based mobile healthcare framework for dyslexic students. The proposed tool was used as a e-learning tool to enhance students learning capabilities. Nepal et.al in (Nepal, 2015) address the requirements for deploying secure hybrid clouds

for Healthcare data. They stress on the need for encryption of PHR in public cloud environments and availability of secure development APIs.

In (Man HoAu, 2017) authors deduce that any public cloud based security model may have inherent threats of trust, security and privacy. They highlight that the complexity of key management in encrypted PHR systems increases for the owners and users resulting in Quality of Service issues. (Zhou, 2015) note that hybrid cloud environment integrating the public and private cloud infrastructure is a more applicable solution for PHR sharing. (Liu, 2017) present a PHR access control scheme allowing access policies to be embedded into PHRs. They conduct simulations studies to verify that the proposed scheme is suitable for mobile-cloud health care. (Zhang, 2017) provide a Quality of Service perspective in Cloud services computing and provide a systematic and practical overview of Quality of Service prediction in cloud and service computing.

Most existing work has adopted ABE (Ostrovsky, 2007) as the cryptographic tool to achieve fine-grained access control in cloud- based PHR systems. The original ABE system supports only monotone access policy and depends on a single private key generator. However, much work has been done in enhancing the ABE scheme. (Man HoAu, 2017) provide a general framework for secure sharing of PHR data. They modify the ABE scheme to encrypt access control information which allows stakeholders to encrypt and decrypt data without losing privacy preserving mechanisms. (Liu, 2018) present a modified Comparison based Encryption technique (CBE). The proposed hierarchical comparison-based encryption (HCBE) scheme incorporates an attribute hierarchy into CBE. The proposed scheme encrypts a ciphertext with a small number of generalized attributes at a higher level rather than many specific attributes at a lower level, greatly improving the encryption performance.

The work presented in this paper addresses the requirements for deploying a secure, low cost low power cluster for storage/ retrieval of personal healthcare records. We provide architecture of a hybrid cloud based PHR system that allows secure access and sharing of PHR based on access control parameters. The proposed security mechanism is based on Attribute based Encryption scheme. Our work is unique in terms of deployment of a power efficient system using a low-cost cloud computing cluster.

7. CONCLUSION AND FUTURE WORK

In this work, we have presented a design and architecture of a hybrid cloud environment for personal health records management using a low cost low power cluster built using two kinds of Single Board Computers. Details for requirements specifications for the proposed system were presented by classifying the requirements at three levels of data handling and data privacy: i) Data acquisition, ii) Data storage in Cloud, iii) Data Processing, analytics and visualization. Based on these requirements, a light weight architecture coupling the public and private cloud implementations was presented. Detailed aspects of encryption mechanisms based on ABE scheme were also presented. A low lost low power Hadoop cluster was deployed using two kind of single board computers, Raspberry Pi 3B and Odroid Xu4. Various experiments were conducted to evaluate the performance of the LoC4 cluster for computation time, storage management and network communications. Response time and task execution time were computed for various sizes of datasets to validate the security mechanism using a virtual machine in the proposed system.

Results from these studies show that a SBC based LoC4 cluster can be successfully deployed in a hybrid cloud environment. Although RPi computers are 50% cheaper than the Odroid Xu-4, the Odroid computer outperform RPi computers in all aspects of performance parameters. Xu-4 performs 3.8 times faster for Task execution times, 2.4 times faster for buffered read and write, 1.8 times faster for network throughput and at least 3 times faster for Hadoop based job executions. While SBC based clusters are energy efficient overall, the operation cost to performance ratio can vary based on the workload compared to a Personal Computer. In terms of power efficiency, for smaller workloads the Xu4 clustered devices performs 1.7 times better compared to RPi devices. Based on these observations,

we conclude that LoC4 cluster is ideal for a hybrid cloud deployment where light weight applications can run on the back end. For heavier workload application, such as big data applications, due to the inefficient capabilities of these devices compared to high end server, the SBC based clusters may not be a performance effective choice. It is however possible to tweak Hadoop configuration parameters to adjust with given resources to improve the overall performance.

We provide response time evaluation experiments to validate the proposed scheme for encryption of data blocks based of ABE scheme. Results from the implementation of ABE based hybrid cloud scheme show on average a 43% improvement compared to AES based scheme for data block sizes of 1.8 MB to 400 MB. The proposed encryption scheme performs 2.3 times faster compared to AES based scheme. The current implementation is limited to xml based data files mimicking a PHR data record. We intend to extend the proposed system to include Apache Hive based Database on the back-end LoC4 cluster with further deployment of MySQL based implementation in the virtual machine running in the public cloud. We also did not consider using secure communication channels. The current system relies on network sockets for listening and responding over TCP/IP protocol. In the future, we intend to incorporate secure and reliable communications mechanisms in the proposed system. We intend to further investigate the effects of various QoS parameters on the communication in the proposed system.

ACKNOWLEDGMENT

This work is partially funded by Research and Innovation Center at Prince Sultan University.

REFERENCES

Abbas, H., Maennel, O., & Assar, S. (2017). Security and privacy issues in cloud computing. *Annales des Télécommunications*, *72*(5-6), 233–235. doi:10.1007/s12243-017-0578-3

Alghabban, W., Salama, R., & Altalhi, A. (2017). Mobile cloud computing: An effective multimodal interface tool for students with dyslexia. *Computers in Human Behavior*, *75*, 160–166. doi:10.1016/j.chb.2017.05.014

Apache Hadoop Framework. (2017). Retrieved from https://hadoop.apache.org/Hadoop

Asija, R., & Nallusamy, R. (2016). Healthcare SaaS Based on a Data Model with Built-In Security and Privacy. *International Journal of Cloud Applications and Computing*, *6*(3).

Au, M. H., Yuen, T. H., Liu, J. K., Susilo, W., Huang, X., Xiang, Y., & Jiang, Z. L. (2017). A general framework for secure sharing of personal health records in cloud system. *Journal of Computer and System Sciences*, *90*, 46–62. doi:10.1016/j.jcss.2017.03.002

Bahga, A. & Madisetti, V.K. (2015), Healthcare Data Integration and Informatics in the Cloud. *Computer*, *48*(2), 50-57.

Basu, S., Karp, A. H., Li, J., Pruyne, J., Rolia, J., Singhal, S., & Swaminathan, R. et al. (2012). Fusion: Managing Healthcare Records at Cloud Scale. *Computer*, *45*(11), 42–49. doi:10.1109/MC.2012.291

Baun, C. (2016). Mobile Clusters of single board computers: An option for providing resources to student projects and researchers. *SpringerPlus*, *5*(1), 360. doi:10.1186/s40064-016-1981-3 PMID:27064532

Diffie, W., & Hellman, M. (2006). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644–654. doi:10.1109/TIT.1976.1055638

Fang, R., Pouyanfar, S., Yang, Y., Chen, S., & Iyengar, S. (2016). Computational Health Informatics in the Big Data Age: A Survey. *ACM Computing Surveys*, *49*(1). doi:10.1145/2932707

Gou, Z., Yamaguchi, S., & Gupta, B. B. (2016). Analysis of various security issues and challenges in cloud computing environment: a survey. In Handbook of research on modern cryptographic solutions for computer and cyber security (pp. 393-419). Hershey, PA: IGI Global. doi:10.4018/978-1-5225-0105-3.ch017

Health Insurance Portability and Accountability Act of 1996. (1996). Retrieved from https://www.hhs.gov/hipaa

Li, M., Yu, S., Zheng, Y., Ren, K., & Lou, W. (2013). Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *IEEE Transactions on Parallel and Distributed Systems*, *24*(1), 131–143. doi:10.1109/TPDS.2012.97

Liu, X., Liu, Q., Tao, P., & Wu, J. (2017). Dynamic access policy in cloud-based personal health record (PHR) systems. *Information Sciences*, *379*, 62–81. doi:10.1016/j.ins.2016.06.035

Nepal S., Ranjan R. & Choo K. (2015). Trustworthy Processing of Healthcare Big Data in Hybrid Clouds. *IEEE Cloud Computing*, *2*(2), 78-84.

Odroid Xu-4 detailed specifications (2017). Retrieved 1 September 2017 from http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825

Ostrovsky, R., Sahai, A., & Waters, B. (2007). Attribute-based encryption with non-monotonic access structures. In ACM conference on Computer and communications security (CCS '07) (pp. 195-203). doi:10.1145/1315245.1315270

Ostrovsky, R., Sahai, A., & Waters, B. (2007), Attribute-based encryption with non-monotonic access structures. In Proceedings of ACM CCS '07 (pp. 195-203).

Page, A., Kocabas, O., Soyata, T., Aktas, M., & Couderc, J. (2014). Cloud-based privacy-preserving remote ECG monitoring and surveillance. *Annals of Noninvasive Electrocardiology*, *20*(4), 328–337. doi:10.1111/anec.12204 PMID:25510621

Page, A., & Soyata, T. et al.. (2015), Visualization of health monitoring data acquired from distributed sensors for multiple patients. In *Proc. IEEE Global Telecommun. Conf.* doi:10.1109/GLOCOM.2015.7417414

Peyton, L., Bindra, J., Baarah, A., Chamney, A., & Kuziemsky, C. (2015). A Bounded Health Information Technology System Design Approach to Support Community-Based Care Delivery. *International Journal of Cloud Applications and Computing*, *5*(1), 32–45.

Qureshi, B. (2014). Towards a Digital Ecosystem for Predictive Healthcare Analytics. In *Proceedings of the 6th International Conference on Management of Emergent Digital EcoSystems (MEDES '14)* (pp. 34-41). doi:10.1145/2668260.2668286

Raspberrypi.org. (n.d.). Raspberry Pi Single Board Computer Model 3B. Retrieved from https://www.raspberrypi.org/products/raspberry-pi-3-model-b

Soyata, T., Copeland, L., & Heinzelman, W. (2016). RF energy harvesting for embedded systems: A survey of tradeoffs and methodology. *IEEE Circuits and Systems Magazine*, *16*(1), 22–57. doi:10.1109/MCAS.2015.2510198

Sun, I., & Reddy, C. (2013). Big data analytics for healthcare. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '13)* (pp. 1525-1525). DOI: doi:10.1145/2487575.2506178

Venkatasubramanian, K., Banerjee, A., & Gupta, S. (2010). PSKA: Usable and secure key agreement scheme for body area networks. *IEEE Transactions on Information Technology in Biomedicine*, *14*(1), 60–68. doi:10.1109/TITB.2009.2037617 PMID:20007032

Wan, J., Zou, C., Ullah, S., Lai, C.-F., Zhou, M., & Wang, X. (2013). Cloud-enabled wireless body area networks for pervasive healthcare. *IEEE Network*, *27*(5), 56–61. doi:10.1109/MNET.2013.6616116

Wang, C., Xu, X., Shi, D., & Lin, W. (2014). An Efficient Cloud-Based Personal Health Records System Using Attribute-Based Encryption and Anonymous Multi-receiver Identity-Based Encryption. In *Proceedings of the Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, Guangdong (pp. 74-81).

Wang, X., Gui, Q., Liu, B., Jin, Z., & Chen, Y. (2014). Enabling Smart Personalized Healthcare: A Hybrid Mobile-Cloud Approach for ECG Telemonitoring. *IEEE Journal of Biomedical and Health Informatics*, *18*(3), 739–745. doi:10.1109/JBHI.2013.2286157 PMID:24144678

Yi, L., Zhang, Y., Ling, J., & Liu, Z. (2018). Secure and fine-grained access control on e-healthcare records in mobile cloud computing. *Future Generation Computer Systems*, *78*(Part 3), 1020–1026.

Zhang, Y., & Lyu, M. R. (2017). *QoS Prediction in Cloud and Service Computing: Approaches and Applications*. Springer. doi:10.1007/978-981-10-5278-1

Zhou, J., Cao, Z., Dong, X., & Lin, X. (2015). PPDM: A Privacy-Preserving Protocol for Cloud-Assisted e-Healthcare Systems. *IEEE Journal of Selected Topics in Signal Processing*, *9*(7), 1332–1344. doi:10.1109/JSTSP.2015.2427113

Zhou, J., Lin, X., Dong, X., & Cao, Z. (2015). PSMPA: Patient Self-Controllable and Multi-Level Privacy-Preserving Cooperative Authentication in Distributed m-Healthcare Cloud Computing System. *IEEE Transactions on Parallel and Distributed Systems*, *26*(6), 1693–1703. doi:10.1109/TPDS.2014.2314119

ENDNOTES

1 https://wiki.gentoo.org/wiki/Sysbench
2 Using sysbench --test=cpu --cpu-max-prime=10000 --num-threads=*n* run
3 https://www.openhub.net/p/fio
4 https://iperf.fr/
5 http://bitspjoule.org/netpipe/

Basit Qureshi obtained BSc and MSc degrees in Computer Science from Ohio University, OH and Florida Atlantic University, FL in 2000 and 2002 respectively. He received PhD in Computer Science from University of Bradford, UK in 2011. He has published over 50 research articles in refereed journals and conferences. Dr. Qureshi received the best paper award in the 6th IEEE Trustcom 2010. His research interests include Trust, Security and Privacy in Low cost and energy efficient systems. He is a member of IEEE, IEEE Computer Society and ACM.