National College of Ireland

Analysis of hybrid encryption in cloud environments for privatization of health data

MSc Research Project
Cloud Computing

Anay Desai
Student ID: X23210125

School of Computing
National College of Ireland

Supervisor: Jorge Mario Cortes Mendoza

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Anay Desai
Student ID:	X23210125
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Jorge Mario Cortes Mendoza
Submission Due Date:	29/01/2025
Project Title:	Analysis of hybrid encryption in cloud environments for privatization of health data
Word Count:	7121
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	29 January, 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Analysis of hybrid encryption in cloud environments for privatization of health data

Anay Desai
X23210125

Abstract

Times have revolutionized the cloud domain with its data storing capabilities and hosting services. This incentivizes the concept of remote storage for total user accessibility. However, this data is not secured and is put to risk for data misuse. This often leads to exposure of user data and breaches the integrity of the system. Moreover, since cloud is an open-source platform; simultaneous services to users often leads to data leaks. To avoid such scenarios; it has become crucial to secure data repositories from third party usage to be protected on the cloud. Therefore, the proposed research thesis aims to maintain the security and confidentiality of the system while deploying a healthcare app on various cloud platforms and leverage Cloud Service Providers(CSPs). A hybrid encryption mechanism is created from a novel approach using Advanced Encryption Standards(AES) and Advanced Encryption with Associated Data (AEAD). Additionally, Amazon Web Services(AWS) along with Continuous Integration and Continuous Delivery(CI/CD) Pipeline is also integrated into the project to ensure security and data integrity such as CodeRun, CodeDeploy and CodeBuild. The time metrics of the text conversion is evaluated, furthermore through case studies and Avalanche score is generated to indicate the strength of the algorithm with results reaching up to 93.8%. MySQL is used in the back end to manage the database and monitor user uploads. The healthcare app is finally deployed on cloud using the multi cloud technique using Microsoft Azure and Elastic Beanstalk.

1 Introduction

Cloud computing is one of the advanced technologies that is primarily used to share resources on a public platform so that user demands could be met efficiently. It is a way of pooling computing resources in such a way that various businesses and application infrastructure tends to come together. This enhances its functionality to store information of multiple users in the cloud environment through a secured manner. High levels of security and confidentiality is maintained so that sensitive information can be stored.

In scenarios of storing patient sensitive data in the domain of healthcare, it is very crucial and important to select the right algorithm whose service could match with the needs of the web application. A commonly used and a recommended practice for the same, is the adoption of encryption techniques. This key in particular is further used to scramble user sensitive data and therefore make it unreadable. This unreadable data is then uploaded on cloud and a decryption key is required to make the data readable. This process

of generating an encryption and decryption key can be done through various methods such as user access controls, Multi-Factor Authentication (MFA) based access control. Both the key can however be generated and accessed only to authorized users. Despite the presence of various encryption algorithms, the developed web app is still vulnerable.

1.1 Research question

The aim of the study is to implement a hybrid encryption mechanism, involving two algorithms that could execute the generation of the hashed key while securing user data on cloud. Therefore, it is necessary that both the algorithms (AES and ChaCha20Ply 1305) are in sync with each other. To fulfill this purpose, AWS Services are integrated with the conceptual working of Continuous Integration and Continuous Delivery Pipeline to ensure a seamless execution of encryption can result in an uninterrupted process of user interaction. Furthermore, the healthcare app is expected to be deployed through a multi cloud environment using MS Azure and Elastic Beanstalk. This leads to the primary research question of the thesis

- What are the modern AES techniques to Enhancing Confidentiality and Data Protection in Multi-Cloud Deployments Using Hybrid Encryption for the healthcare industry

Additionally, a test case scenario is also built by making some changes in the plain text (input) and further converting it into cipher text. This is done to check the strength of the algorithms thus used which eventually leads to the next focus area of the thesis: What impact would the project system undergo in terms of changes being made in the input data and how will that change affect the overall performance of the project?

1.2 Research objectives

Following are the research objectives of the study: a) To evaluate a novel framework by combining a hybrid encryption framework along with AWS services in accordance with the CI/CD Pipeline. b) To deploy the framework using MS Azure and Elastic Beanstalk while maintaining the confidentiality and the integrity of the system.

1.3 Motivation

The motivating factor for the developed project comes from the very fact that there is a need to deploy an automated yet robust system that can pace with the growing evolution of cyber-attacks. Due to its property of being an open-source platform, monitoring a healthcare web app deployed on cloud is mandatory to secure since it is exposed to unauthorized access. While traditional methods to secure patient data are growing; it still faces limitations where complex transactions are involved. In such a scenario, implementing a hybrid model by combining two encryption algorithms with variably small key size appears to be a decent solution. Therefore, adopting this strategy the performance as well as the scalability of the project system is likely to improve in an overall manner. Hence, a combination of two encryption algorithms amalgamated with AWS Services and executed in parallel with CI/CD Pipeline can set a new yet innovative standard to protect patient data on the healthcare web app.

2 Related Work

The process of encryption enables the protection of user data from unauthorized users and is further performed using two types of keys; symmetric key and asymmetric key. The encryption method of an asymmetric key is accomplished using a private key and a public key. On the other hand, symmetric encryption is done using a single key to encrypt and decrypt the data. However, this exchange of key between the user and the recipient must be secured irrespective of the keys thus used. To achieve this security encryption algorithms are used so that user data is not only secured during the transaction of keys; but also during the decryption process.

In recent times, securing this key has remained a major concern since transmission of these keys through an online portal depends on the key size, memory size and the computational complexity thus involved in the process. Cloud has therefore gained massive popularity in accomplishing these goals in a secured manner. This section of the thesis therefore highlights the contribution of various authors in this domain of using encryption keys and algorithms to protect user data transmission on cloud.

2.1 Encryption Algorithms

In a system proposed by authors Shukla et.al, (2020) a combination algorithm of AES and Elliptic Curve Cryptography (ECC) was used to increase system security. Shamir's Secret Sharing (SSS) key was generated so that distribution and management of user data could be done without a trusted Centre. Despite the fact of a well-organized secured system, the overall computational cost and time complexity were highly introduced.

Yahia et.al, (2021) used the combination of above-mentioned algorithms along with Blowfish. This however reduced the time complexity and made the algorithms more relatable to secure user data on cloud. The used algorithms also improved the efficiency and integrity of the system and also avoided user conflict by properly channelizing resource allocations so that each user can access cloud services. Data accessibility in this was also managed by the service provider. The author also calculated the avalanche score in the research so that the impact of plain text and its related changes in the block size could be measured. The strength of the algorithms such as ECC and Rivest-Shamir-Adleman (RSA) were measured by Qazi, R.; Khan et.al, (2019). The algorithms were combined over 256 bytes of data flow and performance amongst encryption algorithms was calculated. The implementation showed a higher side of efficiency produced by using ECC to that of Rivest-Shamir-Adleman (RSA). In addition to providing secured service to user data, the overall size of the data was also drastically reduced. This in turn minimized the computational complexity and time complexity thus involved. The program code was however run on JAVA.

Chen et.al, (2021) created a layering method was used to implement encryption on user data. The first layer was divided into small sections of data bytes which were eventually encrypted on the basis of key size. On the other hand, the second layer was partitioned into various elliptical curves such as A0, A1, A2, A3,. . . ..An. Both the layers thus used ensured high levels of security and minimized data loss. Breakdown of the data was done using asymmetric techniques wherein larger chunks of data were managed using two different keys. ECC was simultaneously used and the final data was deployed on cloud. Arockia, P et.al, 2017 focused their work in the services provided by CSP's. It

was observed that securing user data and protecting its privacy were the two main important tasks to be carried out while deploying user data on cloud. Hence, he used AES and ECC in combination so that the overall transmission capacity of the information could be increased and a classification framework could be built to ensure more security on cloud. The expected deployed model was therefore trustworthy and was implemented with less computational power.

Pothireddy et.al, (2024) proposed the implementation of Fully Homomorphic Encryption (FHE) and Secure Hashing Algorithm-3 as the encryption algorithm. His focus was to increase the security, privacy and integrity of the data deployed on cloud. A differentiating factor in his work was to encrypt the user data and deploy on cloud without the further need to decrypt it. This process eliminated the scenario of unauthorized susceptibility that could have resulted into a breach of the system. In a different work proposed by (Kamble et.al, 2024) he addressed the relevance of homomorphic encryption wherein multiple clouds were used by the author to deploy his work. This was done to protect user data by storing it across various cloud environments. The concept however, experienced certain disadvantages such as handling large complexity of data and generation of different keys.

Alabdulatif et.al, (2017) introduced a novel method to secure user data by performing clustering methods using the FHE algorithm. This clustering ensured that the data is fully secured on cloud. As per the results of this framework, it was observed that the distributed framework of FHE significantly reduced the overall time complexity involved in the system. Additionally, a Brakerski, Gentry and Vaikuntanathan (BGV) scheme was also adopted and MapReduce was used to improve system efficiency. This model generated an accuracy of more than ninety percent and thereby protected user data on cloud environments.

2.2 Encryption models of ChaCha20-Poly1305

Thi-Thanh-Dung Phan et.al, (2016) contributed their study in the implementation of securing user data through FGPA protocol and further combining it with AES-Counter with CBC-MAC(CCM) algorithm. This combination provided higher efficiency with minimal time involved as compared to other combination of encryption algorithms. The author also provided a different angle which stated various security requirements of the model and also introduced a few techniques that could later protect the system from different breaches and attacks. Despite introducing new techniques and strategies of the same; an emergence of wireless network body areas was observed which required low areas to consume power in the encryption process. Hence, the author shifted his focus to low areas of AES-CCM implementation using FGPA. The only drawback of this proposal was that it could not be implemented on hardware platforms since the resource allocation for its execution was not capable enough.

Fabrizio De Santis et.al, (2017) proposed the encryption mechanism of ChaCha20-Poly1305 to implement user authentication in IoT applications. A ChaCha stream of cipher text was used and Poly1305 was designed to be the cryptographic algorithm by Daniel J. Bernstein. The aim of the author was to ensure that high security of use data could be achieved that could be deployed on various software platforms. In another work Sadiq Aliyu Ahmad et.al, (2019) presented a review of hybrid cryptography with 12 tabular surveys being made. The aim of this research was to comprehend the amalgamation of cryptography and encryption on cloud.

2.3 AWS Services and Encryption

Mishra et al., (2022) examined the methods in cloud computing that caters to securing user data without compromising on its performance whereas Saeed et al., (2019) implemented AWS Services to ensure security and privacy measures of the system. Additionally, Talha et al., (2020) evolved various algorithms that addressed the issues of networking with high security. For this, they emphasized on increasing the overall quantity of user data and later uploading it on cloud. Boomija and Raja, (2023) targeted the healthcare sector that offered a solution to monitor and store user data on cloud. For this they combined Secure Partially Homomorphic Encryption (SPHE) with AES. The overall efficiency of the system was drastically changed on this implementation. Mishra, (2023) deployed user data on cloud using AWS Services by including Elastic Beanstalk (EBS), Elastic File System (EFS) and further leveraging it through AWS Direct Connect, Snowball and Storage Gateway.

All these studies were attempted with the intention to overcome challenges and difficulties that were connected to cloud. The algorithms and strategies thus used provided high levels of user security with low computational complexity.

2.4 Comparative Analysis Table

The summary of the analysis of the table is done in Fig 2.

Approach	Algorithm	Metric	Method	Problem Resolved	Dataset	System	Ref
MPC	NRGD	A, EXECUTION TIME	SSS, CKKS	Data Privacy, Data Sharing	ECO-Dataset	Simulation, AWS	1
FHE, MPC	LR, BGD, NRGD	PRECISION, RECALL, TIME	CKKS, SSS	Computation on Encrypted Data	Simulated Dataset	Simulation	2
SHE	RING-LWE	THROUGHPUT, TIME	FV	Latency operations, Computational Inefficiency	Smart Grid Application Dataset	AWS	3
HE	AES, RSA	SECURITY, ACCURACY, TIME	SYMMETRIC	Security, encryption efficiency, Privacy	Various Datasets	Azure	4
FHE	FWC	EXECUTION TIME	HDFS	Data Privacy, Efficient Data Clustering	UC Irvine Machine Learning Repository (Electricity Consumption Dataset)	Azure	5
PHE	AES	THROUGHPUT, DOUBLE ENCRYPTION	Data Privacy	Medical Dataset	-	AWS, Simulation	6
CGFHE	INTEGRITY, FUNCTIONALITY	SYMMETRIC	Data privacy, integrity, availability	Simulated Data	-	AWS	7
LHE	DFHE, FCM	TIME, SENSITIVITY	MPHS	Data Security, Scalability and Efficiency	Intel Berkeley Research Lab Sensor Dataset	GCP	8
HE	AES, DES	A.SECURITY, TIME	Data Security, Integrity, Data Sharing	AW, SG, CP	-	AWS and Azure	

Table 1: Comparison of various approaches and their performance metrics.

2.5 Research Gap

The traditional process of a cryptography technique involves the storing of user data in an encrypted form and getting it deployed on cloud. User data is later decrypted using a secret key. Since data is exposed to an open-sourced platform, generation of this secret key is the most challenging task in the entire process. There are high chances of data loss and breach of key features. This results to be a grave issue and here emerges the need to develop a method that could store this key and protect it from unauthorized access. Therefore, the presented report addresses this issue that takes place in the healthcare sector to secure

patient sensitive data. For this purpose, two cryptographic techniques are used to enhance confidentiality and security of the system deployed on cloud. The study's key point is to build an infrastructure by integrating a hybrid mechanism with AWS Services. The hybrid mechanism makes use of two encryption algorithms; Advanced Encryption Standards (AES) and ChaCha20-Poly1305. Hence, the aim of the project is to install the hybrid encryption model by using AWS Services such as that of Elastic Beanstalk and finally deploying it on Flask. The pipeline integration on the backend is simultaneously performed using Continuous Integration and Continuous Delivery Pipeline which in turn tends to align with the AWS standard rules. A novelty created at this stage is the deployment of same project model using Microsoft Azure. The entire project system thus deployed tends to leverage Python as the programming language and increases the potential of securing user data on cloud storage systems. In summary, the presented thesis aims to highlight the implementation of two encryption mechanisms (AES and ChaCha20-Poly1305) deployed on two AWS cloud infrastructures (MS Azure and Elastic Beanstalk) by running the web app in the backend using Continuous Integration and Continuous Delivery Pipeline. The thesis therefore intends to significantly improve the efficiency of the healthcare app without compromising on the security of user data.

3 Methodology

This section of the thesis briefs on the research design that has been adapted to deploy healthcare data on cloud. Through the literature survey thus conducted, it can be observed that certain limitations and restrictions do occur in the conventional research flow. One of the primary concerns was to manage the costs associated with computational overhead that eventually increased with larger datasets. Response time for the same was also observed to be high. Such issues resulted for a compromise being made in the security of user data which is deployed on cloud. In order to overcome such issues and limitations, the thesis presents a hybrid approach that combines the implementation of a dual encryption algorithm. MySQL is used as the backend database to monitor user data. Encryption is performed on the sender and the receiver's side with the exchange of the secret key through mail ID. Additionally, CodeRun, CodeDeploy and CodeBuild along with CI/CD Pipeline is used to deploy the web app. It is important to note here that such large amount of user data is securely stored on cloud by using a multi-cloud approach. For this purpose, MS Azure and AWS Service based Elastic Beanstalk (EBS) is used. The experimental setup thus created tends to meet user demands without system lag and delays. The sections below highlight the conceptual working implementation of the methodologies thus involved.

3.1 The phases

3.1.1 Encryption

With advancements being made in the digital world, its crucial to store user data in a secured manner. This is done to prevent unauthorized access of data from hackers. A set of standard rules are followed so that privacy violations are not made. Encryption is a simple process of converting a data in a readable format to a format which is unreadable. This unreadable data format is called as the cipher-text.

3.1.2 Decryption

The process of encryption serves the purpose of forming a barrier against unauthorized access of user data to breaches. On one hand, encryption converts readable data to

unreadable data; decryption is the act of restoring unreadable data to its original form. This transformation of data is however performed by making use of certain keys. Such keys are responsible to reverse the cryptographic processes and thereby generate the plain-text from cipher-text (Sinaga, M. D, 2018)

3.2 Keys

Keys are generally crucial elements of a cryptographic process since they are responsible to encrypt and decrypt the entire process. Keys are majorly of two types.

3.2.1 Symmetric Keys

Symmetric key cryptography makes use of a single key to perform the process of encryption and decryption. The key is commonly known as the private key. This key is further shared between the user and the recipient so that communication entities could be formed. Its implementation is however faster in comparison to asymmetric key cryptography; since only one key is involved (Jankowski et.al, 2011).

3.2.2 Asymmetric Keys

Asymmetric Key cryptography makes use of two keys which are generated separately on the user and the recipient side. The key is commonly known as a public key. Both the communicating entities can make use of this key to decrypt the message. However, the key faces two major challenges during the process of execution. Former being the process of key distribution and latter being, digital signatures. Since the process involves the presence of two keys on both the sides; encrypting or decrypting user data becomes impossible by knowing only one key. However, a mathematical function is also used for conversion of plain-text to cipher-text and vice versa.

3.3 Algorithms Used

3.3.1 AES

It is one of the most commonly used techniques in the field of cryptography. Its implementation process is however based on several mathematical operations which are executed in the form of block-ciphers. AES is capable to handle and manage 16 bytes of data which is approx- imately 128 bytes of plain-text. This plain-text is represented in the form of a 4 x 4 matrix which operates in rounds over the bytes. The number of rounds is completely reliable on the length of the key size. Primarily three varying key sizes are used to encrypt and decrypt using AES (128 bytes, 192 bytes, and 256 bytes). These key sizes are responsible to decide the number of rounds the algorithm would make to complete plain-text conversion. AES is a form of symmetric key cryptography since it used only one key on both the ends of the communication parties. The strength of AES algorithm can be calculated through various statistical analysis which ensures that cryptography is implemented in alignment with cloud storage.

3.3.2 ChaCha20-Poly1305

ChaCha20-Poly1305 is an encryption algorithm which is associated with the AEAD algorithm. The process tends to combine ChaCha20 stream cipher-text along with Poly1305 text so that authentication can take place. It is a fast delivery software algorithm developed to enhance the overall performance of the system. Its implementation is therefore faster than AES-GCM. It further takes input in the form of 256-byte key and a 96byte key to encrypt the plain-text. In the next stage, a cipher text expansion of 128 byte long key size is made and the plain text is XORed. A cipher-text is then generated which is used to authenticate user data. This way enables the authentication of one user at a time.

3.4 Methodologies Used

3.4.1 AWS

The Amazon Web Services provides a range of massive services which can be access by the user with pay as per use pricing model. Its service ranges from providing resource storage to computational power and further hosting them on web apps deployed on cloud. A variety of tools are also used in the process which enables flexibility in the system thus created. Keeping in mind AWS always comes up with its own networking infrastructure with already established datacenters. It also has a global infrastructure which provides virtual services like Amazon Virtual Private Cloud (VPC), Amazon Elastic Cloud Compute(EC2) and Amazon Simple Storage Service (S3). Dynamic support is also provided by AWS which helps to scale the web app and further take care of issues related to load balancing. The overall AWS architecture ensures a seamless user interaction with the web app so that a versatile cloud computing platform is built.

3.4.2 CI/CD

CI/CD is one of the most important and commonly used automating practices by several developers. It allows them to commit to their codes without the need to simultaneously deploy them on the central database. The usage of CI/CD inhibits the process and reflects the code automatically in the central repository. Hence the need to update the code is eliminated. This central repository is often observed to be GitHub. Additionally, there are is a provision for automated process that follows the concept of static analysis tools to review the code during the testing phase. Integrating these tools not only helps to improve the quality of software development; but also reduces the overall time complexity of the system. The working of the pipeline is shown in Fig 1.

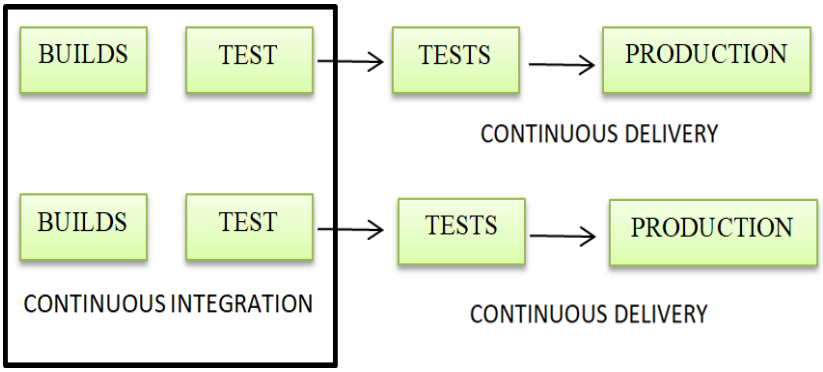


Figure 1: CI/CD pipeline

Another additional advantage of using CI/CD is that it is capable to detect bugs and errors in the system and further notifies the developer on its occurrence. This leads to elimination of system lags and reduces the response time of the software by D. Ståhl, et.al,(2014).

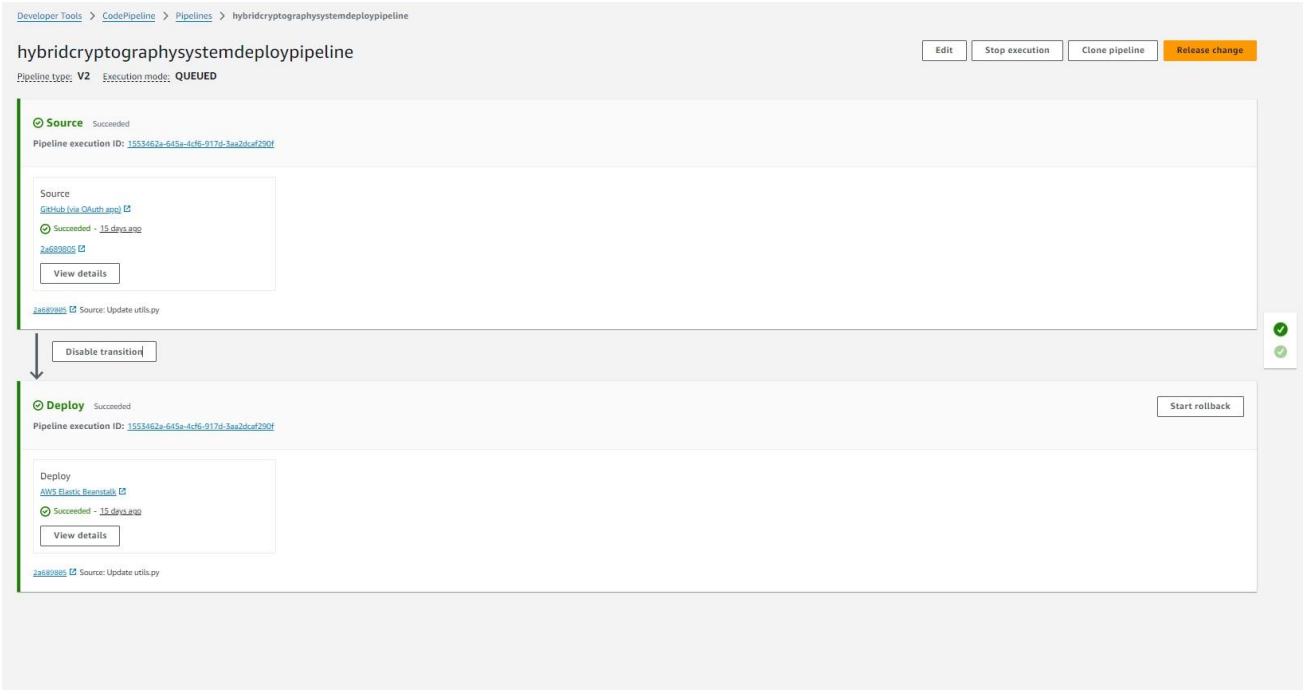


Figure 2: working Pipeline of the Project

3.5 Deployment Framework Used

3.5.1 Microsoft Azure

Microsoft Azure is a cloud computing service provided by Microsoft which enables the user to deploy his web app on cloud. It's an open-source platform and makes the entire process a flexible app building task.

3.5.2 Elastic Beanstalk

EBS is also a service provided by AWS that enables the developer to run his codes without the need to worry about back end issues and load balancing. It can also be used to deploy web apps on cloud portals.

The working architectural diagram of the research is shown in Fig 3

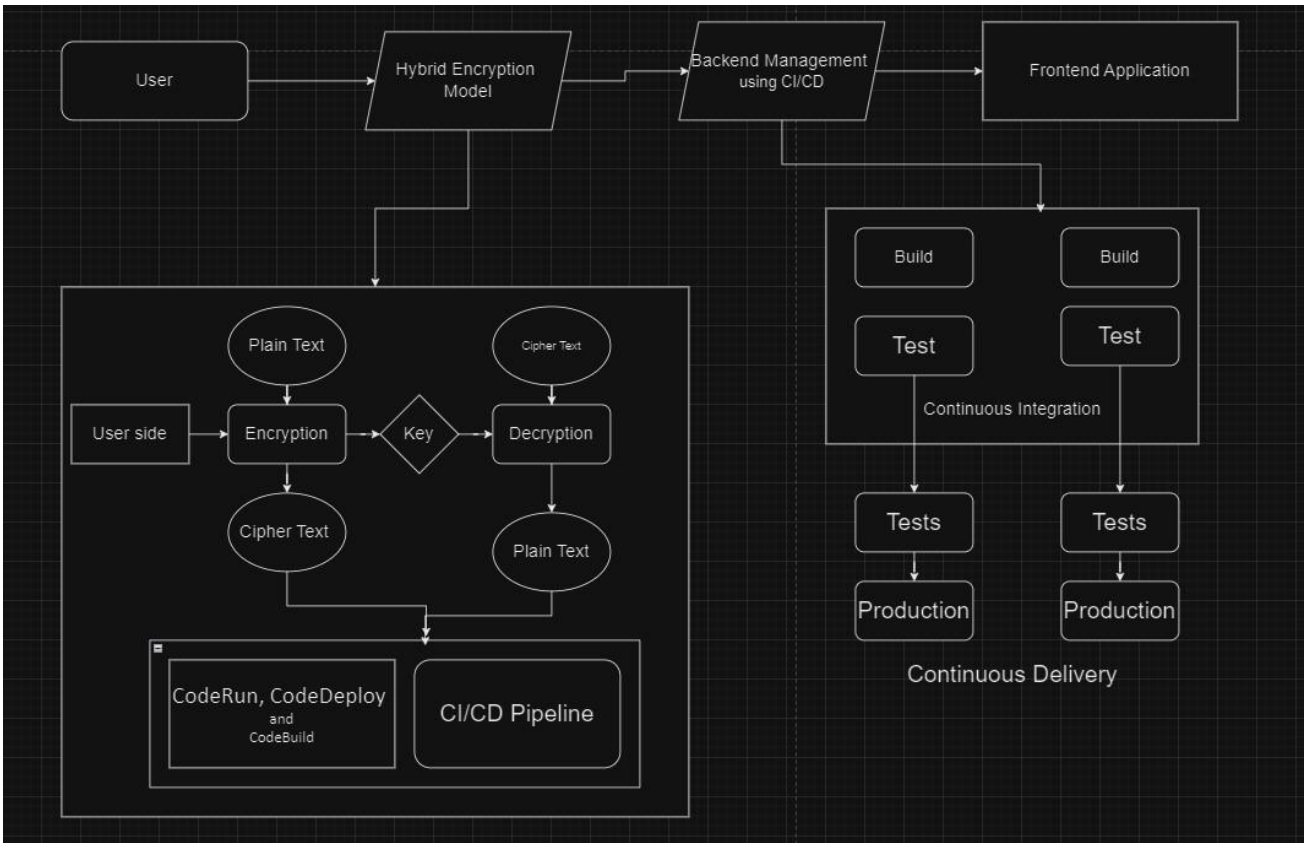


Figure 3: Architectural Diagram

4 Design Specification

4.1 Overview of the Workflow

The work in this thesis proposes to deploy a healthcare system on cloud, using Flask as the GUI interface. A hybrid method of using AES and ChaCha20- Poly1305 is implemented. The encryption algorithms are primarily meant to im-prove the overall security and integrity of the system model. The scheme is further based on a multi-cloud system that uses two cloud services (MS Azure and EBS) to deploy the project.

To enable a smooth interaction of the users on the web portal, the system design integrates various critical functionalities. The user can initially register him on the web app by providing his mobile number and mail ID. The credentials entered by the user in this stage are safely stored on the backend.

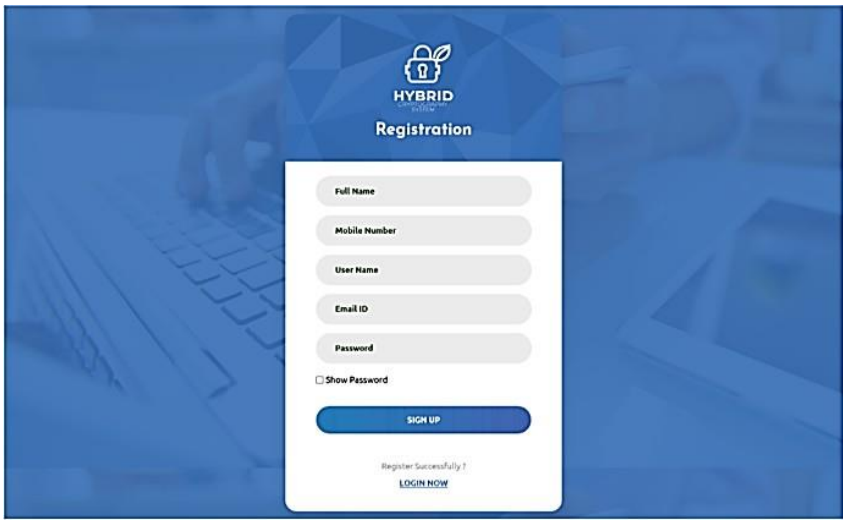


Figure 4: Login credentials of the user

After a successful login he is then directed to his dashboard. The Hybrid Cryptography system allows him to add health care data including his age, blood pressure levels, and cholesterol present in blood and heart rate.

Figure 5: User adding his healthcare data on web portal

The user then submits this information. Meanwhile in the backend during this procedure; a series of encryption stages along with algorithms are implemented. A hybrid approach of AES and ChaCha20-Poly1305 are triggered to collect user data and convert them into respective cipher-texts takes place. The process of key generation also occurs and is further shared between the user and the recipient. This ensures that a safe upload of user data is done on cloud. It additionally secures all the transactions therefore being made on the portal.

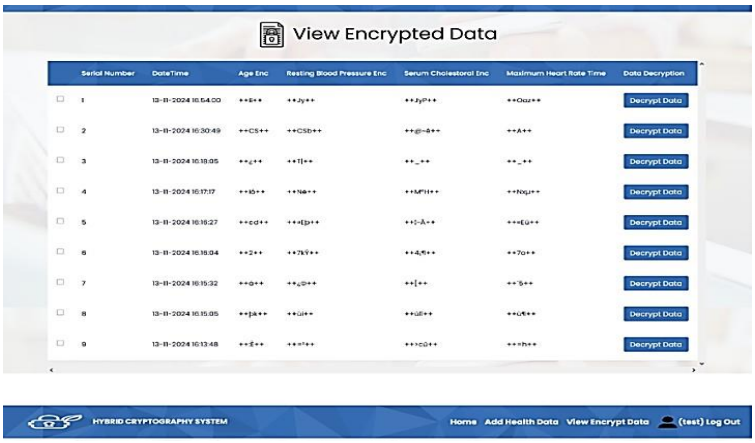


Figure 6: Encrypted data

Figure 7: Enter key to view Decrypted data

MySQL is used on the backend to trigger queries and the admin later send the generated key to the user through his registered mail ID. The user logs into the portal through his credentials and uploads the data. The decryption key is further sent to the registered user on his registered mail ID through which the conversion of cipher text to plain text occurs. An Simple Mail Transfer Protocol(SMTP) is used in this process so that the mail could be sent. The system design of the model carefully adheres to providing utmost security to user data wherein the user can download the plain text file only on receiving the key. Simultaneously a series of AWS Services and integration of the CI/CD Pipeline is observed in the project. Finally the healthcare app is deployed in the multi cloud environment by using MS Azure and Elastic Beanstalk. The entire execution of healthcare on cloud prioritizes to secure user communication and interaction being made on the web portal. The workflow below depicts the system design of the thesis.

The system design above provides an exact blueprint for deploying the web app on a multi cloud environment by using a hybrid approach. The AWS Services used in the design, enables to focus on the interface to be developed so that a smooth functionality is executed on Flask. The design also highlights the emphasis being made on the user end, the management of files for seamless encryption followed by the deployment process.

5 Implementation

The data implementation of the thesis revolves around using a hybrid encryption method that uses AES and ChaCha20-Poly1305 as algorithms to store user data on the healthcare app. The approach utilizes the property of speed and efficiency from AES while using the robust property of ChaCha20-Poly1305. A detailed process of key exchange takes place so that the system is benefitted to improve the encryption process of the research. The encrypted AES key is securely passed to the recipient on his mail ID using the SMTP protocol. A simultaneous encryption of ChaCha also takes place. Once the recipient decrypts the data using the key; he can then access his information. This execution of two algorithms in a hybrid module confirms and ensures that the communication between the two entities is not breached. It also tends to provide a balance between security and performance of the system model.

The architecture of the system is however designed to protect user data and focus on operational reliability. All the functionalities of securing the system and monitoring user data are performed efficiently in a user friendly environment. AWS Services along with CI/CD Pipeline are used in combination to build, run and deploy the project by using CodeBuild, CodeRun and CodeDeploy respectively.

For the purpose of deployment, a multi cloud concept is used and the application is hosted by Microsoft Azure and AWS Elastic Beanstalk. This gives the cloud a proper architecture to enhance its security, reliability and scalability. Azure provides a seamless workflow of resource allocation that can handle change of loads whereas; EBS ensures that adequate amount of computer provisioning is done so that the developer can run codes in a hassle free environment. Finally, MySQL is used in the backend to trigger queries and manage the database. This DBMS thus used guarantees that the deployed architecture is strong and thereby protected against breaches and hacking activities.

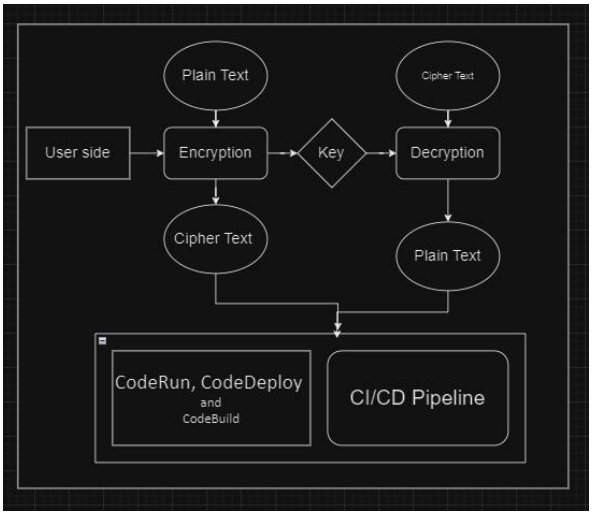


Figure 8: Encryption Design

6 Evaluation

This section of the thesis provides a complete detail on the implementation process executed on the back end while the system is deployed on cloud and is accessed by the user. Explanation of the same is divided into three fragments; with each fragment contributing to briefing of the process to induce comprehended results.

Avalanche Effect

In the field of cryptography, the Avalanche effect is generally linked so as to how a text would react during the encryption and decryption process in terms of mathematical functions. A minor change being made in the input text would have a massive impact on the output being generated. This property is termed as the avalanche effect; which tends to have a significant impact in the domain of cryptography. For instance, a small change in one byte of the input message would alter the entire encrypted message and therefore secure it from attacks by third parties. Hence, it is considered to be as the most desirable property that an algorithm must inhibit during the process of encryption. The execution of the same is conventionally done on block ciphers by making use of hash functions. In scenarios of block ciphers with high quality, a small change made in the plain text tends to have drastic change in the cipher text thus produced. Figure 8 below depicts the illustration of Avalanche Effect.

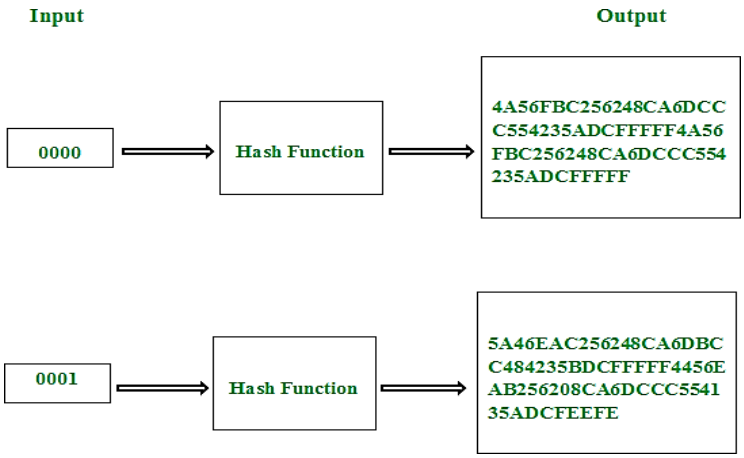


Figure 9: Avalanche Score

In cases where the hash functions do not produce avalanche effect to a specific and desired degree; the plain text is more likely to be attacked by a cryptanalyst. This indicates that the built encryption algorithm is not strong enough to secure the data on cloud. Therefore, avalanche effect can also be used to check the strength and weakness of an algorithm Avalanche effect can however be calculated as:

Avalanche Effect = Number of flipped bytes in cipher text / Number of bytes in ciphered texts

6.1 Case Study 1: Change of 1 byte

This case study enlightens the process of hybrid encryption that takes place on the back end. Once the user logs into the healthcare app through his credentials, he is then directed to mention four user attributes such as age, blood pressure levels, cholesterol levels and heart rate. The input values are taken and termed as “original data”. Two test case scenarios are expected to occur. The first scenario involves the direct conversion of plain text to cipher text by using hash function and later calculating its encryption and decryption time. The second test case scenario involves a change being made in one byte of the input data and further converting the plain text to cipher text by using hash function and later calculating its encryption and decryption time. This analysis helps to understand the strength of the algorithm by changing one byte in the input data. The analysis is however comprehended through the Avalanche Effect. Both the test case scenarios are explained below:

6.1.1 Original Data

The experimental analysis for this thesis is based on the encryption process that takes place on the backend. Generation of the keys and the process of encryption and decryption are monitored. The case studies below briefs on the time required to execute encryption process, decryption process, time required to execute the same, generation of keys, conversion of plain text to cipher text, usage of hash function followed by the avalanche effect.

User Attribute	Plain Text	Hashed Key	Cipher Text	Encryption Time	Decryption Time
Age	55	QEkvpVpqKG4TwRCjB3b6Q==	bG9/ZW0xaXBzdWI1ZX09SXVDVA==	1.96	2.2
Blood Pressure	85	Rm9vQmFyQmFzZTY0QWzwaGEM=	Y2lwaGVyVGVzdFBeWxvYWQwOTIg=	1.5	1.78
Serum Cholesterol	230	SkVUo2blbkFYQ3N2MjByNzg5Mjk=	dGlhpc0lZRkjnclwdGVkVGdV4dA==	2.1	2.34
Maximum Heart Rate	99	SGFzaGVLS2V5RXhxbXBsXBZzXkIy=	Zm9yVXNIQFZvZXN0dWRSFGfOYQ=	1.76	1.9

Table 2: Original Data Encryption and Decryption

The table above explains the user attributes thus entered by the user and the corresponding plain text to it. Hashed key is the link that converts the plain text to cipher text. Once the hash key is generated; the number 55 in column 2 row 2 is converted to its respective cipher text which is in turn inaccessible by the attacker. The encryption and decryption time required to perform the above conversions is depicted in the table.

6.1.2 Altered Data with change in one byte

User Attribute	Plain Text	Altered Text	Hashed Key	Cipher Text	Encryption Time	Decryption Time
Age	55	56	SGFzaGVLS2V5RXhxbXBzXBYzXkIy=	Zm9yVXNIQFZvZndWRSFGOYQ=	1.6	1.84
Blood Pressure	85	86	RGVtbGluRGFYZ2lhVz5mcYGVgYQ=	Y25kc2VwUHVHUHV2c2Dy==	1.35	1.56
Serum Cholesterol	230	232	QUFWTmpJRXJ2WZAbmIJGzUlCE=	ZW5jVmRlUkGFc3dmcnEdICFEz==	2.1	2.35
Maximum Heart Rate	99	98	UERyQUldURgyIXNGljA2NZoVZ==	ZGVybY29pZGV6INJXNcmVf==	1.84	2.12

Table 3: Altered Data by One Byte and Avalanche Effect

The table above explains the scenario wherein one byte from the original data of the plain text is changed. The changed plain text is then used to generate the respected hashed key. In scenario below, 56 is used for hash key generation. Using this key, the

cipher text of its respective number is generated. The encryption and decryption time required to perform the above conversions is depicted in the table.

6.1.3 Calculation of Avalanche Score

Once the encryption and decryption time for original data and changed data is calculated; the avalanche score can further be determined to analyse the strength of the system. By combing tables the generated avalanche score is as follows:

User Attribute	Original Data	Changed Data by One Byte	Avalanche Score (%)
Age	55	56	92.85714286
Blood Pressure	85	86	89.28571429
Serum Cholesterol	230	231	88.285782857
Maximum Heart Rate	99	98	92.85714286

Table 4: Avalanche Effect Calculation

6.2 Case Study 2: Change of 2 bytes

This case study will follow the same procedure as Case study 1 but the score will be calculated between 2 byte changes.

6.2.1 Original Data

The contents are replicated from the 1st case study to show the randomness of the algorithm.

6.2.2 Altered Data with change in two bytes

Table below explains the scenario wherein two bytes from the original data of the plain text are changed. The changed plain text is then used to generate the respected hashed key. In scenario below, 56 is used for hash key generation. Using this key, the cipher text of its respective number is generated along with their time metrics.

User Attribute	Plain Text	Altered Text	Hashed Key	Cipher Text	Encryption Time	Decryption Time
Age	55	62	QkFTRTY0S2V5VGVhcyZWVGdXJ0aG	cGFzdWxnRTduUjdpbmRldWlzZX	1.45	1.65
Blood Pressure	85	74	QWxpZVZHW5cmF0ZU2haV2FbmM=	c2pweXBtRkd2cmdolGlxXaRoS2V	1.53	1.8
Serum Cholesterol	230	221	UmFuZG90UGlsXGVUZjV1QjVjV2hJcJ	ZW5jb2RlRUlCc3dmcnQdMNVNJB	2.03	2.16
Maximum Heart Rate	99	78	TC9pRGVyaGxYZ1Z6bNtbCMlNtyXA=	cm9zGVVGZzN0cmNlUoNVUn==	1.56	1.84

Table 5: Altered Data Encryption and Decryption

6.2.3 Calculation of Avalanche Score

Once the encryption and decryption time for original data and changed data is calculated; the avalanche score can further be determined to analyse the strength of the system. By combing tables 7, 8, 9 and 10, the generated avalanche score is as follows:

User Attribute	Original Data	Changed Data by two Bytes	Avalanche Score (%)
Age	55	62	90.85714286
Blood Pressure	85	74	87.35159429
Serum Cholesterol	230	221	89.104522857
Maximum Heart Rate	99	78	93.842314286

Table 6: Avalanche Effect Calculation

User Attribute	Hashed Key	Cipher Text	Avalanche Score (%)
Age	oQ+aqttFmU4vScda4NXD+w==	7VIxOFzzVF0MIVzxIc/08jOiUiA=	90.85714286
Blood Pressure	oQ+aqttFmU4vScda4NXD+w==	7nQjOBfk4kR2zI/H3cqIAQ36QBE=	87.35159429
Serum Cholesterol	5frxGr2yXEZKpU9PVk+9A7m4KJ8hgIMbmY+frFxoU3M8=	7IAxOKV5fvANqxN44mG964weOWw=	89.104522857
Maximum Heart Rate	3P4t4/vH1GJ9rR8zXq3zYQ0gA6+bXw7IN6RVjxKH07o=	sRzYg5tXL/R+oQ9NqFUy8M7kPXTVo3Fq5jWZgFYt7Fc=	93.842314286

Table 7: Avalanche Effect Calculation

The avalanche score is calculated using the following script.

```
17  binary1 = '/UQCorC+w4WBZ3hSO1MPKA=='
18  binary2 = '8GeHbrAA3sOKJ6xU5lM94g=='
19
20
21  avalanche_score = calculate_avalanche_score(binary1, binary2)
22  print('Avalanche Score:', avalanche_score)
23
```

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  SEARCH ERROR  DEBUG CONSOLE

WebApplication>C:/Users/Administrator/anaconda3/envs/hybridcryptographyenv/python.exe c:/U
ers/Administrator/Desktop/hybrid_cryptography_system_in_cloud/WebApplication/avalanche_eff
ct.py
Avalanche Score: 97.82608695652173
```

Figure 10: Avalanche Score

6.3 Load Balancing in Cloud Environment

This section of the chapter focuses on the backend run being performed in the project. With multiple users being available on the portal at the same time tends to impact the load balance of the system thus deployed. Monitoring and balancing this load is necessary so that system delays and failure can be avoided. For this purpose, an open source load testing tool named as “locust” is used in Python language. It is a software testing tool which is responsible to target the occurrence of load in the website and later configure the web UI in real time. The figure below illustrates the locust dashboard for MS Azure

Figure 11: Locust for MS Azure

The figure above indicates three graphs with respect to encryption being performed on user data, deployed on MS Azure. The first graph represents the number of requests handled by the system in one second. The second graph represents the response time given by the system to attend the user requests. While the third graph represents the number of users being available on the web app. The characteristic feature of the system to balance load; however leads to the rise of three parameters. CPU Utilization, Network In and Network Out. Defining these parameters is necessary so that the performance of the CPU can be calculated in a multi cloud environment. The three parameters thus extracted and deployed on MS Azure and AWS are explained further.

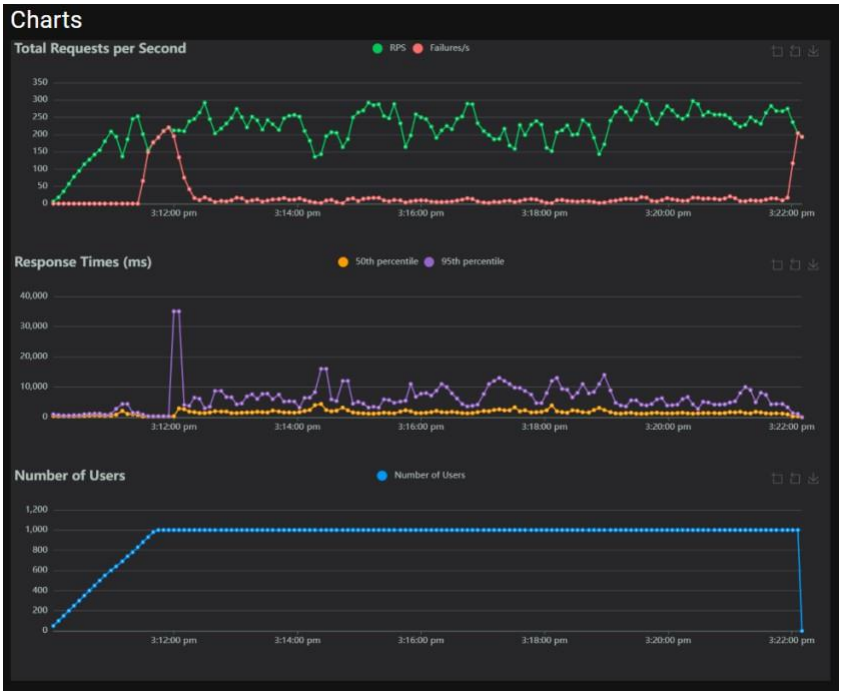
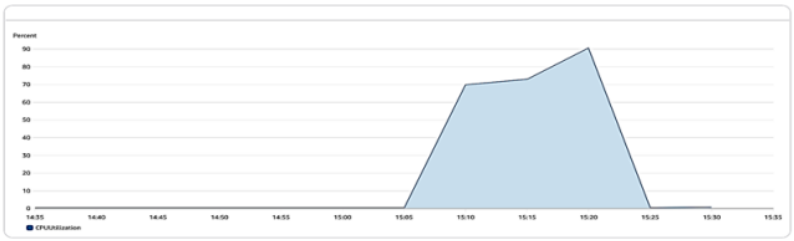


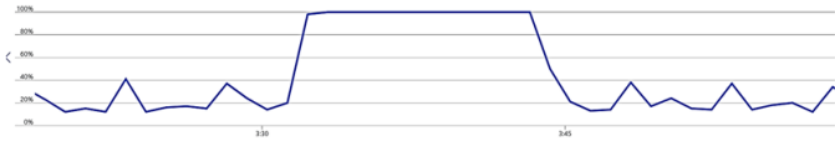
Figure 12: Locust for AWS

6.3.1 CPU Utilization:

This metric is responsible to reflect the load occurring on the system's CPU while the web app is deployed on the cloud. A high CPU Utilization is often an indication that the system is overloaded and over-burdened and therefore cannot generate effective results. This metric highly depends on the number of users accessing the web app at a particular instance, the requests received from the user every second, the complexity of the tasks thus involved and the configuration of the system. Hence, Locust is used so that an even form of load distribution can be done so that a significant amount of computational overhead is reduced. The figures below illustrate the CPU Utilization when the web app is deployed on MS Azure and AWS.



(a) AWS

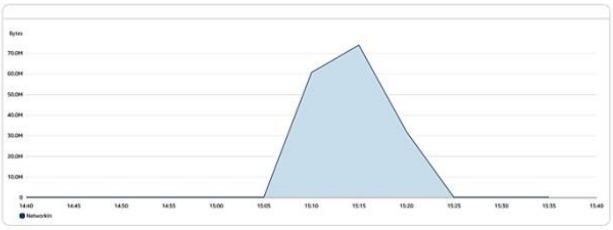


(b) MS Azure

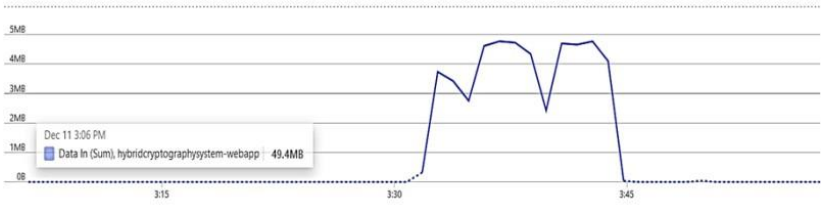
Figure 13: CPU Utilization on Multi Cloud

6.3.2 Network In

This is an important metric which is responsible to measure the total amount of data thus received by a system network over a period of time. Hence, it is a very crucial tool to analyze the load occurring on the system so that the performance of the network can be calculated. The network In parameter heavily depends on the number of concurrent users along with its request payload. The figures below illustrate the Network In parameter being deployed on MS Azure and AWS.



(a) AWS

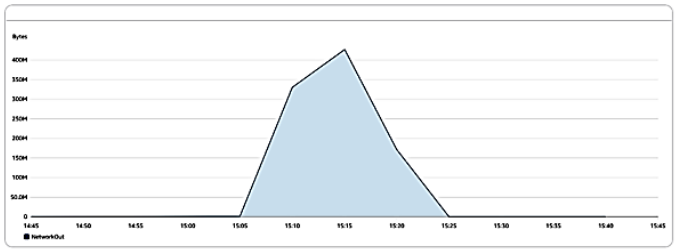


(b) MS Azure

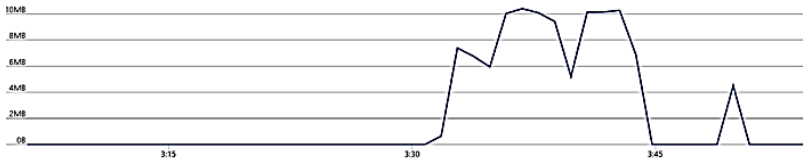
Figure 14: Network In parameter on Multi Cloud

6.3.3 Network Out (bytes)

This term is an indication of the amount of data being sent to other systems over the network and hence represents the outbound traffic which is made from the server in response to the requests thus made. This parameter is important to analyze since determines the range of bandwidth used by the server and therefore optimizes the payload size so that the system performs better. Network Out majorly depends on the number of concurrent users and the size of response payloads thus given to each server. Figure below illustrates Network Out in AWS and MS Azure.



(a) AWS



(b) MS Azure

Figure 15: Network Out parameter on Multi Cloud

Through this section, it can therefore be concluded that deployment of healthcare app on AWS to encrypt user data is more beneficial. Since CPU Utilization and Network Out traffic were better handled in AWS during peak hours, the encryption performed on the back end and further deployed on AWS serves the final objective of the system i.e to deliver consistent performance under payload.

7 Conclusion and Future Work (final recap)

The focus of the research is to deploy a healthcare app on cloud and creating a novel encryption by combining 2 encryption types. AES and ChaCha20Poly1305 are used in combination to secure user data and generate respective keys to safeguard the data The procedure of securing user data on cloud is massively significant due to the incorporation of AWS Services along with CI/CD Pipeline. The final deployment of the web app is done using a multi cloud environment through MS Azure and Elastic Beanstalk. The highlight of the project implementation is in the experimental analysis wherein the encryption and decryption time along with the avalanche effect is calculated. Avalanche effect is a measure of how strong a system is and can be determined by calculating the number of changes made in the input given to the system. As per tables 6 and 4 thus generated in chapter 6 of the thesis; it can be observed that the Avalanche score were higher when changes were made in two bytes. This indicates that the hybrid encryption algorithm was stronger in case study 2 rather than case study 1. Due to this factor the second case study becomes more difficult for any cryptanalyst to break into the code. Table 11 below illustrates the Avalanche score generate in both the test cases.

Avalanche score with 1 byte change	Avalanche score with 2 byte change
92.85714286	90.85714286
89.28571429	87.35159429
88.285782857	89.104522857
92.85714286	93.842314286

Table 8: Case Study Evaluation.

However the existing limitation of the system restricts its execution to only numbers; wherein the user can enter the numerical factors in accordance to age, blood sugar levels, cholesterol levels and heart rate. Another limitation of the system is the slow encryption time observed during the testing phase.

In the future, the current research thesis can be extended to a hybrid encryption in multi cloud environment on texts, numbers as well as images. Furthermore, different AWS Services can be incorporated into the system such as AWS Lambda; so that the developer need not worry to run codes during the detection of errors and bugs.

REFERENCES

1. Shukla, D. K., Dwivedi, V. K., & Trivedi, M. C. Encryption algorithm in cloud computing. Materials Today: Proceedings, 2021, 37, 1869-1875.
2. Yahia, H. S., Zeebaree, S. R., Sadeeq, M. A., Salim, N. O., Kak, S. F., Adel, A. Z., ... & Hussein, H. A. Comprehensive survey for cloud computing based nature-inspired algorithms optimization scheduling. Asian Journal of Research in Computer Science, 2021, 8(2), 1-16.
3. Khan, I. A., & Qazi, R. Data security in cloud computing using elliptic curve cryptography. International Journal of Computing and Communication Networks, 2019, 1(1), 46-52.
4. Chen, Y.; Liu, H.; Wang, B.; Sonompil, B.; Ping, Y.; Zhang, Z. A threshold hybrid encryption method for integrity audit without trusted center. *J. Cloud Comput.* **2021**, *10*, 3
5. Arockia, P.; Dharani, N.; Aiswarya, R.; Shailesh, P. Cloud data security using elliptic curve cryptography. *Int. Res. J. Eng. Technol.* **2017**, *4*, 32–36
6. Kamble, A., Jiet, M.M. and Puri, C. (2024) 'Homomorphic Encryption and its Applications in Multi-Cloud Security', in 2024 International Conference on Inventive Computation Technologies (ICICT). 2024 International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal: IEEE, pp. 1493–1499
7. Phan, T.-T.-D., Hoang, V.-P., & Dao, V.-L. (2016). An efficient FPGA implementation of AES-CCM authenticated encryption IP core. 2016 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS)
8. De Santis, F., Schauer, A., &Sigl, G. (2017). ChaCha20-Poly1305 authenticated encryption for high-speed embedded IoT applications. Design, Automation & Test in Europe Conference & Exhibyteion (DATE), 2017. doi:10.23919/date.2017.7927078
9. Ahmad, S. A., &Garko, A. B. (2019). Hybrid Cryptography Algorithms in Cloud Computing: A Review. 2019 15th International Conference on Electronics, Computer and Computation (ICECCO). doi:10.1109/icecco48375.2019.9043254
10. Alabdulatif, A. et al. (2017) 'Privacy-preserving Data clustering in Cloud Computing based on Fully Homomorphic Encryption
11. Mishra, S., Kumar, M., Singh, N. and Dwivedi, S., 2022, May. A survey on AWS cloud computing security challenges & solutions. In *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 614-617). IEEE
12. Saeed, I., Baras, S. and Hajjdiab, H., 2019, February. Security and privacy of AWS S3 and Azure Blob storage services. In *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)* (pp. 388-394). IEEE

13. Talha, M., Sohail, M. and Hajji, H., 2020. Analysis of research on amazon AWS cloud computing seller data security. *International Journal of Research in Engineering Innovation*, *4*(3), pp.131-136
14. Boomija, M.D. and Raja, S.K., 2023. Securing medical data by role-based user policy with partially homomorphic encryption in AWS cloud. *Soft Computing*, *27*(1), pp.559-568
15. Mishra, P., 2023. Advanced AWS Services. In *Cloud Computing with AWS: Everything You Need to Know to be an AWS Cloud Practitioner* (pp. 247-277). Berkeley, CA: Apress
16. Halder, S. and Newe, T. (2022) 'Enabling secure time-series data sharing via homomorphic encryption in cloud-assisted IIoT', Future Generation Computer Systems, 133, pp. 351–363
17. 'Data Security in Cloud Environment by Using Hybrid Encryption Technique: A Comprehensive Study on Enhancing Confidentiality and Reliability' (2024) International Journal of Intelligent Engineering and Systems, 17(2), pp. 159–170
18. Turan, F., Roy, S.S. and Verbauwhede, I. (2020) 'HEAWS: An Accelerator for Homomorphic Encryption on the Amazon AWS FPGA', IEEE Transactions on Computers, pp. 1–1
19. A. N. Bhat and R. Kumar, "Efficient Hybrid Encryption Algorithm for Securing Data in Cloud Environment," Apr. 18, 2024
20. Alabdulatif, A. et al. (2017) 'Privacy-preserving Data clustering in Cloud Computing based on Fully Homomorphic Encryption'
21. S. Murthy and C. R. Kavitha, "Preserving Data Privacy in Cloud using Homomorphic Encryption," in 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India: IEEE, Jun. 2019, pp. 1131–1135
22. M. M. Potey, C. A. Dhote, and D. H. Sharma, "Homomorphic Encryption for Security of Cloud Data," Procedia Computer Science, vol. 79, pp. 175–181, 2016
23. Sinaga, M. D., Sembiring, N. S. B., Tambunan, F., &Sianturi, C. J. M. (2018). Hybrid Cryptography WAKE (Word Auto Key Encryption) and Binary Caesar Cipher Method For Data Security. 2018 6th International Conference on Cyber and IT Service Management (CITSM)
24. Jankowski, K., & Laurent, P. (2011). Packed AES-GCM Algorithm Suitable for AES/PCLMULQDQ Instructions. IEEE Transactions on Computers, 60(1), 135–138
25. D. Ståhl and J. Bosch, "Modeling continuous integration practice differences in industry software development," Journal of Systems and Software, vol. 87, pp. 48–59, 2014