National College of Ireland

Enhancing Cloud Security: A Hybrid PGP-Inspired Encryption System Using ECDH and AES-GCM

MSc Research Project
Cloud Computing

Siddharth Manik
Student ID: x23289678

School of Computing
National College of Ireland

Supervisor: Shaguna Gupta

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Siddharth Manik
Student ID:	x23289678
Programme:	Cloud Computing
Year:	2024-2025
Module:	MSc Research Project
Supervisor:	Shaguna Gupta
Submission Due Date:	02/01/2026
Project Title:	Enhancing Cloud Security: A Hybrid PGP-Inspired Encryption System Using ECDH and AES-GCM
Word Count:	7322
Page Count:	24

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:	24th January 2026

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):	

Enhancing Cloud Security: A Hybrid PGP-Inspired Encryption System Using ECDH and AES-GCM

Siddharth Manik
x23289678

Abstract

The rapid growth of cloud-based web applications has been increased concerns regarding secure data transmission, key management and system performance under concurrent type of workloads. There are some traditional encryption approaches which prioritise either strong security or high performance by resulting in trade-offs that are unsuitable for dynamic cloud environments. This study addresses the need for a balanced, scalable and secure encryption framework capable of operating in cloud-native architectures. To solve this problem there is a Pretty Good Privacy (PGP) inspired hybrid encryption system which was designed and implemented using Elliptic Curve Diffie–Hellman (X25519) for secure session key exchange and symmetric encryption with AES-GCM, DES included for legacy comparison. The system has been deployed as a stateless web application on AWS. Concurrent workload evaluation has been conducted with the help of Locust to simulate real-world cloud usage. Experimental results shows that the AES-GCM consistently outperformed DES in terms of latency.

Keywords: Hybrid Encryption, PGP-Inspired Framework, Cloud Security, Elastic Beanstalk

1 Introduction

In this chapter, the background and motivation of secure data handling in cloud-based environments were presented and the shortcomings in current encryption methods were established. It outlined the research problem solution, objectives and given a PGP-inspired hybrid encryption framework to strike a balance between security, performance and scalability of cloud-based systems.

1.1 Background

The web-based applications have become the main medium of sharing data, communicating, and storing information in a cloud where information security has become a critical issue Chatterjee et al. (2023). Cloud computing gives users an opportunity to save and access information remotely via the internet and therefore, cloud computing presents sensitive data to risks of unauthorized access, interception and data breaches Singh and Bhushan (2025). The basic cryptographic process of ensuring the security of such data is through the encryption of data (such as the Advanced Encryption Standard - AES) which uses a single secret key to encrypt and decrypt information, providing high performance

and low security on the distribution of keys.

1.2 Motivation

As the use of cloud-based applications is gaining pace, data confidentiality is under consideration when it is transmitted and stored. Conventional encryption systems are unable to provide a good balance between high security and reasonable performance in dynamically congested cloud environments. In addition, a number of systems are based on the practices of key management that are static which make it easier to compromise the keys. This research is motivated by the fact that there is a need to design and test a hybrid encryption system. The proposed method incorporates the asymmetric key exchange and effective symmetric encryption and is expected to improve the security, forward secrecy and evaluate the real world performance of the solution in a cloud environment.

1.3 Problem Statement

However, even with the availability to use more complex cryptographic algorithms, most cloud applications are not based on a well-structured and performance sensitive encryption system that will allow secure key exchange, scalability, and efficiency which can be assessed. Current solutions are either security-centric but are unaware of computational overheads or are speed-centric at the cost of cryptographic security. This study deals with the issue of creating a hybrid encryption framework that can safely handle session keys, run stateless in the cloud and carry out a quantitative performance and load based analysis to analyze the trade-offs among security, efficiency and scalability.

1.4 Research Question

What impact does a PGP-inspired hybrid encryption framework have on security and performance in cloud and how does the use of key exchange with AES-GCM and DES encryption affect measurable outcomes under concurrent cloud workloads?

1.5 Problem Solution

To address the identified challenges of secure key management, scalability and performance in cloud environments, this study has proposed a PGP-inspired hybrid encryption framework designed for cloud-native applications. The solution integrated Elliptic Curve Diffie–Hellman (X25519) for secure and ephemeral session key exchange with high-performance symmetric encryption with the help of AES-GCM whereas DES is included solely for legacy comparison.

1.6 Research Objectives

There are 2 research objectives which includes:

1. To design and implement a PGP-inspired hybrid encryption framework that uses asymmetric key exchange with symmetric encryption algorithms (AES-GCM and

DES) in a cloud-based environment which secure session key establishment and data confidentiality.

2. To evaluate the performance of modern and legacy encryption techniques by measuring key generation time, encryption and decryption latency, resource utilization and throughput within a Web based application which has been deployed on AWS infrastructure.

1.7 Research Contribution

The study contributes a research-based PGP-inspired hybrid encryption system to cloud-native applications with the combination of ephemeral X25519 key exchange and authenticated encryption by using the AES-GCM. It shows an AWS-based stateless encryption service and empirically compares the current (AES-GCM) and the old (DES) algorithms with concurrent workloads, which emphasize on the performance, scalability, and forward secrecy in actual clouds.

1.8 Research Scope

The proposed study focuses on the design, implementation, and testing of a hybrid encryption system to transmit data in web applications on clouds and ensure their safety. These features are the X25519-based ephemeral key exchange, HKDF-based session keys, AES-GCM and DES encryption, AWS-based deployment and performance under Locust synthetic concurrent workloads, and are evaluated in terms of latency, throughput, resource consumption and the avalanche effect.

1.9 Limitations of this Study

There are some limitations of this study. The workload proposed is tested in a controlled cloud platform using simulated workloads instead of a real-world production traffic. The system does not include enhanced key management services or security modules only concentrating on the performance of encryption and decryption. Also, the test is restricted to a set of cryptography algorithms and AWS services and this might influence the applicability of the results to other cloud systems or encryption programs.

1.10 Structure of the Study

This report has structured into seven chapters. Chapter 1 introduces the research background, motivation, problem statement, objectives and research questions. Chapter 2 presents a good review of related work by focusing on cloud security challenges and existing cryptographic techniques. Chapter 3 details the research methodology, system design, encryption workflow and experimental framework. Chapter 4 describes the system architecture and implementation which includes AWS deployment. Chapter 5 presents the implementation details. Chapter 6 evaluates the system through performance, security and load-testing experiments. At last Chapter 7 concludes the study and shows limitations and future works.

2 Related Work

The chapter has surveyed the literature available on the problem of cloud security and methods to apply cryptographic tools in symmetric, asymmetric, and hybrid encryption. It found some essential research gaps in terms of the actual cloud implementation, contemporary key exchange schemes, and simultaneous workload analysis, which precondition the proposed research.

2.1 Web Cloud Security and Data Protection Challenges

2.1.1 Growth of Web Cloud Applications

The development of web cloud applications has altered the way organizations provide their services, store data and even have contacts with their users Vellela et al. (2023). Everything, including e-commerce and e-learning, data analytics and enterprise management as well, now depends on a web-based platform, hosted on a cloud infrastructure, by businesses and individuals. The reason why this has moved towards online operations is because cloud technologies are scalable, flexible, and cost effective, and users have the ability to access resources anywhere using web browsers. Web applications in the context of cloud computing can be integrated as a way of facilitating real-time collaboration and constant availability in order to improve the productivity of the users as well as the business efficiency Ajayi (2025). Nevertheless, since these systems deal with sensitive user information, including personal information, financial information, and intellectual property, the matter of security has become a core issue. Cloud-hosted web services due to their distributed nature are vulnerable to various threats such as unauthorized access, data breach, and attacks Dawood et al. (2023) based on network. Advanced encryption and authentication systems are needed to maintain the confidentiality of data, integrity and availability of data across dynamic and shared cloud environments.

2.1.2 Web Data Transmission and Security Issues

Web data transmission forms the basis of cloud-based communication whereby there is constant exchange of information among clients, servers and databases over the public networks Shinde et al. (2024). This continuous flow of data brings about possible vulnerabilities which may be used by way of interception, modification, or accessing the data without proper authorization. Although such secure protocols as HTTPS and SSL/TLS are commonly used to ensure the safety of web traffic Kumar et al. (2024) they can not completely reduce the risk of such advanced attacks as man-in-the-middle intrusions, session hijacking, or data injection. A variety of web apps also rely on third-party APIs and microservices, which provide more vulnerabilities that attackers can use Jayalath et al. (2024).

2.2 Cryptographic Techniques in Cloud

2.2.1 Symmetric Encryption

The analyzed literature works reflect the dynamic environment of cloud computing security and encryption algorithms, the question of the compromise between performance,

scalability, and data confidentiality. Although Basco et al. (2025) made a comparative analysis of Data Encryption Standard (DES) and Advanced Encryption Standard (AES), it was based more on secondary data and not on an experiment, and it concluded that AES is a better protection tool because of its strong key structure and unrestricted to brute-force attack. Continuing on AES, Sarkar et al. (2024) considered its internal structure and phenomenon of error propagation, which showed that although AES is almost immune to traditional cryptographic attacks, small errors in the transmission may influence the reliability of the system, as well as the necessity of much stronger error-handling systems. Similarly, Tajudeen et al. (2025) aimed at enhancing the security of multimedia messages by combining AES with other cryptographic schemes to defeat the vulnerabilities of AES to brute force and the high costs of RSA to computation and discovered that hybrid AES models can increase the speed of performance and data protection. To elaborate, Bauskar (2024) performed an experimental analysis of RSA vs. AES and a hybrid RSA-AES model in Mathematica, and found that the AES encryption and decryption times were the fastest, and hybrid model was the best trade-off between speed and security, albeit with small data size. In the meantime, Revathy et al. (2025) examined the flexibility of AES to mobile, IoT, and cloud environments, where it was noted to be structurally efficient in terms of efficiency and scalability, but still requires empirical investigation in the context of actual systems with limited computational capabilities. To supplement the studies in algorithms, Mahaboob Basha et al. (2023) provided a more extensive view on cloud security, which touched upon issues such as unauthorized access, data breaches, and encryption, focusing on AES as a secure means of guaranteeing confidentiality but not experimenting with it technologically. In a new direction, Chinnam and Sambana (2024) suggested AI-enhanced hybrid encryption architecture based on Blowfish algorithm to overcome the real-time cloud vulnerability as per confidentiality, integrity, and authenticity with conceptual strength, but weaker experimental validation. Lastly, Triple DES (3DES) was proposed in Awadh et al. (2024) to enhance the protection of big data due to the advantages it has over current frameworks, namely, it generates better protection with less computational time than current encryption systems, such as AES. Taken collectively, these studies indicate that although AES is the key element in the world of modern encryption, new hybrid and AI-based models, including but not limited to RSA AES, AES steganography, and AI Blowfish, are showing promising developments in balancing speed, scalability and resiliency. Nonetheless, the majority of the studies are either theoretical or simulation-oriented, and there is an urgent necessity to implement encryption in real-life and conduct experimental testing on large scale to ensure that it is efficient in various cloud and IoT systems.

2.2.2 Asymmetric Encryption

In the current cloud and distributed computing scenario, asymmetric encryption is critical towards assuring data confidentiality, authentication and integrity. Several researches have been done on improvements to conventional systems like RSA of public key in order to overcome the performance, scaling and security issues. First study given by Elumalai and Muruganandam (2024) who gave Secure and Efficient Data Storage and Retrieval (SEDSR) algorithm that incorporates the RSA (4096-bit) algorithm along with a symmetric key algorithm to ensure compact and scalable security in the untrusted cloud environment, which reduced the time spent on encryption and decryption by 1.7 and 1.5 seconds, respectively, and additionally enhanced throughput by 34%. Applying this to

the concept of multi-cloud environments, Manjyanaik and Mohanty (2024) proposed an Improved RSA (IRSA) methodology to secure data on several cloud environments such as AWS, Azure and Google Cloud. The IRSA ensured a faster encryption and decryption of prime factors (103 ms and 110 ms) than the current schemes by avoiding the process of prime factor decomposition, including AWS Identity and Access Management (IAM) to ensure secure access, but was dependent on cloud-specific tools. In the meantime, Patel et al. (2023) conducted a comparative study of RSA and Elliptic Curve Cryptography (ECC) in fog computing and determined that ECC provides the same security with reduced key sizes and increased efficiency in resource-constrained IoT systems. Moreover, Rakhra et al. (2024) introduced a superior digital signature verification system based on asymmetric encryption and streamlined key allocation to guarantee data authenticity and non-repudiation in cloud platforms and provided faster verification with reduced computational overhead. On top of this, Serengil and Ozpinar (2025) presented LightDSA, a hybrid digital signature library, which supports RSA, DSA, ECDSA, and EdDSA, which can be configured to a flexible curve form and provide useful information of RSA trade-offs between efficiency and security as indicated in Table 1. Together, these papers are indicative of the ongoing development of asymmetric encryption- hybrid RSA-based to flexible and efficient signature systems- based cryptography has shown great advances in achieving secure, scalable, and performance-based cryptographic solutions, but there are still issues of interoperability, deployment in the real world, and optimization of computation.

Table 1: Summary Table

Study	**Purpose**	**Proposed Approach**	**Challenges Addressed**	**Key Results**	**Limitations**
Basco et al. (2025)	To enhance multimedia message security using combined cryptographic methods.	AES combined with other cryptographic techniques for multimedia encryption.	High computational cost of asymmetric cryptography and vulnerability of AES to brute-force attacks.	Combined methods improved multimedia data protection and reduced computational cost.	Does not specify optimal hybrid structure; may increase system complexity.
Sarkar et al. (2024)	To improve message security using hybrid encryption for multimedia systems.	Combination of AES with additional algorithms for better performance and security.	Balancing speed, computational complexity, and robustness.	Hybrid approach offered stronger protection than single algorithms.	May increase resource usage and key management overhead.

Study	Purpose	Proposed Approach	Challenges Addressed	Key Results	Limitations
Tajudeen et al. (2025)	To explore efficient AES-based encryption for multimedia security.	Enhanced AES combined with auxiliary security mechanisms.	Ensuring security without increasing computational overhead.	Hybrid AES yielded better security and reliability over single encryption.	Lacks real-world scalability validation and practical deployment testing.
Bauskar (2024)	To evaluate performance of RSA, AES, and a hybrid RSA–AES method for cloud data.	Implemented RSA, AES, and hybrid encryption using Mathematica with datasets (64–176 bytes).	Balancing encryption speed and security strength.	AES was fastest; hybrid balanced speed (0.10–0.20 ms) and security.	Limited to small data sizes; lacks energy and scalability analysis.
Revathy et al. (2025)	To analyze AES performance in various environments (mobile, IoT, cloud).	Performance evaluation of AES across 128-, 192-, and 256-bit key sizes.	Maintaining efficiency in constrained environments.	AES proved highly secure and adaptable for high-speed and IoT systems.	No hybrid comparison or alternative algorithms analyzed.
Mahaboob Basha et al. (2023)	To review cloud security mechanisms and data protection strategies.	Overview of AES encryption and data confidentiality techniques in cloud systems.	Managing privacy, authentication, and access control in cloud computing.	Identified AES as the most used and effective symmetric encryption method.	Lacks experimental validation and implementation details.
Chinnam and Sambana (2024)	To address cloud infrastructure leaks using AI-enhanced security.	Hybrid Encryption Algorithm integrated with Blowfish for cloud protection.	Securing personal and real-time cloud information effectively.	Improved confidentiality, integrity, and data recovery in cloud setups.	No performance data; AI integration not deeply detailed.
Awadh et al. (2024)	To secure big data in cloud environments, especially healthcare data.	Triple Data Encryption Standard (3DES) applied to cloud-based data protection.	Overcoming inefficiency, data insecurity, and third-party dependence.	Enhanced security with less computational time versus intelligent frameworks.	Limited to healthcare data; scalability in other sectors untested.

Study	Purpose	Proposed Approach	Challenges Addressed	Key Results	Limitations
Elumalai and Muruganandam (2024)	To improve secure data storage and retrieval in untrusted cloud environments.	SEDSR combining RSA (4096-bit) with symmetric encryption.	Overcoming inefficiency in single-key encryption systems.	Reduced encryption (-1.7 s) and decryption (-1.5 s) time; 34% throughput improvement.	May face scalability issues with large datasets due to RSA complexity.
Manjyanaik and Mohanty (2024)	To enhance multi-cloud security and data control.	Improved RSA (IRSA) enabling segmented cloud encryption.	Managing fragmented data and security across multiple clouds.	Encryption-decryption times of 103/110 ms for 100 MB.	Dependent on cloud-specific tools, reducing portability.
Patel et al. (2023)	To compare ECC and RSA efficiency in cloud and fog computing.	Analytical comparison of ECC vs. RSA for IoT and fog networks.	Handling low-latency requirements and limited device resources.	ECC provided equal security with smaller keys and faster computation.	Theoretical study; lacks real-world validation.
Rakhra et al. (2024)	To strengthen digital signature verification in cloud computing.	AES integrated with optimized key distribution.	Reducing computational load and improving authentication resilience.	Improved verification speed and scalability.	Limited to controlled environments.
Serengil and Ozpinar (2025)	To design a flexible digital signature library.	LightDSA supporting RSA, DSA, ECDSA, and EdDSA.	Overcoming fixed-curve limitations.	Balanced security and performance with open-source flexibility.	Real-world deployment not fully explored.

2.3 Research Gaps Identified

Although existing studies given by Shen (2023) has examined symmetric and asymmetric encryption techniques in cloud environments where so many gaps remain. Previous works are either theoretical, simulation-based or limited to small datasets which lacks real-world cloud deployment and concurrent workload evaluation. There are some studies which use modern elliptic curve–based key exchange with authenticated encryption in a stateless cloud-native architecture. Also comparative empirical analysis between modern encryption (AES-GCM) and legacy algorithms (DES) under identical type of cloud workloads is largely absent. Performance metrics like CPU usage, memory consumption, throughput and scalability under concurrent users are rarely evaluated together. This

study solves these gaps by implementing and experimentally validating a PGP-inspired hybrid encryption framework which has been deployed on AWS using load testing and performance analysis.

3 Methodology

The chapter introduced experimental research design and the methodological framework that would be used in the application and assessment of the hybrid encryption system. It described the cryptographic process, key exchange procedures, encryption schemes and load testing plan employed to guarantee controlled performance and scalability investigation.

3.1 Research Design and Framework

The study follows an experimental and system-oriented research design by focusing on the design, implementation and evaluation of a hybrid encryption framework which has inspired by PGP. The study aims to enhance data confidentiality, integrity and performance in cloud-based environments by combining asymmetric and symmetric cryptographic techniques.

The methodology has been divided into four logical stages:

1. Secure key generation and exchange
2. Asymmetric data encryption and decryption
3. Cloud-based system implementation
4. Performance and scalability evaluation

This design allows controlled experimentation, accurate performance measurement and direct comparison between modern encryption (AES-GCM) and legacy encryption (DES-CBC).

3.2 Threat Model and Security Assumptions

The current study takes a network-based threat model of web applications hosted on the cloud that uses untrusted networks. The opponents are assumed to be able to engage in passive eavesdropping, man in middle, replay and session compromise attacks. With AES-GCM authenticated encryption and ephemeral X25519 key exchange, these threats are avoided that guarantee confidentiality, integrity, and forward secrecy at concurrent workloads of clouds.

3.3 Key Generation Phase

In case of each encryption request, a receiver key pair is created with Curve25519 (X25519). The generation of the private key is done by the X25519PrivateKey.generate() function which generates a related public key that will be needed in order to communicate safely. The duration of this key generation process is stored in high-resolution timers which are

developed using time.perf_counter_ns. It is a measurement that allows to accurately assess the computational cost of key creation. The algorithm is a simulation of the PGP-style key generation and the guarantees of new cryptographic content are created during each encryption session thus guarding the session level security isolation.

3.4 Ephemeral Key Exchange (ECDH)

An ephemeral key pair is created by the sender to generate a secure session key to use in every encryption request. With the ECDH there is a common secret is calculated between the ephemeral private key of the sender and the public key of a receiver. The overhead in terms of execution time of the ECDH operation is measured in order to analyze the performance overhead of secure key exchange. This method gives forward secrecy, such that loss of long term keys does not impact on previous sessions. It also provides defense against compromised keys and can offer the PGP-style session isolation by having cryptographic contexts independent of each other.

X25519 was selected for ECDH due to its performance, security and suitability for cloud-native systems. The framework generates fresh ephemeral key pairs per request which ensures session-specific keys derived via HKDF. This design provides forward secrecy of long-term keys does not expose past session data, aligning with secure, stateless cloud encryption requirements.

3.5 Session Key Derivation

The common secret obtained as a result of the ECDH key exchange is then converted to a symmetric session key with the help of the HMAC-based Key Derivation Function (HKDF). Under AES mode, HKDF generates a 256-bit symmetric key which is composed of 32 bytes and is viable in secure encryption. DES mode with 8-byte key size The mode uses a 64-bit symmetric key generated to facilitate a comparison of the encryption of old-time encryption. This critical derivation algorithm guarantees that the symmetric session key is deterministically derived throughout the shared secret and the cryptographic separation in between the key exchange and data encryption phases.

3.6 PGP with AES-GCM

PGP using AES-GCM is used as the main mode of encryption in the proposed system. In this mode, asymmetric cryptography is employed to ensure the safe set up of a mutual session key followed by the application of symmetric data encryption through AES-GCM algorithm. AES-GCM is chosen as this algorithm fulfils the authenticated encryption, and the system could provide the data confidentiality, integrity, and authentication in one operation. The mode allows the safe processing of sensitive cloud data and also the accurate measurement of encryption and decryption time. The implementation is a contemporary and safe PGP-based encryption strategy which can be applied in a cloud setting.

AES-GCM provides encryption and authenticity, utilising counter-mode secrecy in conjunction with authentication in a Galois field. An authentication tag is created during encryption and authenticated during decryption to ensure both integrity and authenticity.

Whereas DES-CBC does not provide any mechanism of integrity checks and, therefore, undetected manipulation of ciphertext is possible, making AES-GCM a better tool to use in secure cloud communication.

3.7 PGP with DES

PGP using DES is applied in the secondary mode of encryption to compare with the older one. Just like the AES-GCM mode, asymmetric encryption is done to generate a session key that is used to encrypt using the DES algorithm. DES is a block chaining mode and needs padding to align the blocks. This mode has been added to compare the difference between the performance and resource usage of modern and legacy encryption methods. Despite the acknowledged security constraints of DES, the usage of this system enables the study of the execution time, throughput and systems behavior to be compared under the same conditions in the presented framework.

3.8 Load Testing using Locust

In order to test scalability and robustness, the system is tested by using Locust to model a number of concurrent users that are accessing the system in concurrent mode. Randomized plaintext messages of 20 to 500 characters are created in order to simulate more realistic encryption loads. AES and DES workloads have separate Locust scripts to make sure that each encryption mode is fairly and completely evaluated independently. Encryption POST requests are sent by each virtual user to the /encrypt endpoint and the full encryption process is initiated. At load testing, the important performance metrics are monitored such as request success rate, average response time, load throughput and stability of the system under concurrent encryption load.
This workflow shows with plaintext input, secure key generation and ECDH-based session key derivation which is followed by PGP-inspired encryption using AES-GCM or DES as shown in Figure 1.

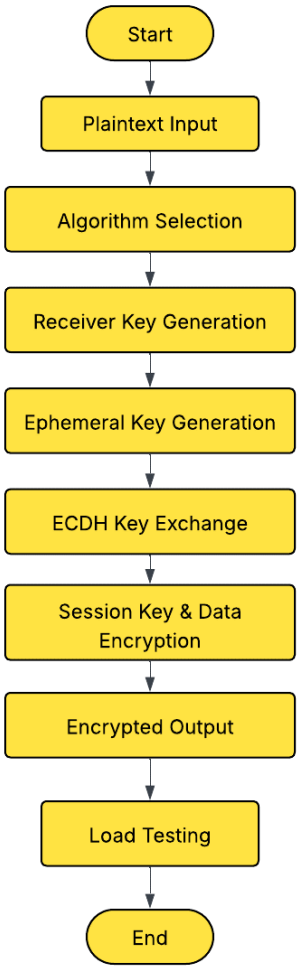

Figure 1: Workflow Diagram

4 Design Specification

The chapter outlined the design of architecture and algorithmic specification of the proposed PGP-inspired hybrid encryption structure. It also rationalized the choice of cryptography methods and described ethical issues and data utilization, which guaranteed a secure, responsible, and cloud-native system design.

4.1 System Architecture

Figure 2 shows the system structure for applied Data Security on Cloud by Implementing Hybrid Cryptography with PGP-Based Encryption. This architecture is multi-leveled, and uses AES-GCM, DES and PGP to encrypt and decrypt files securely. This starts with the sender selecting a random symmetric key for use by AES-GCM for the purpose of encrypting the file. The file is then encrypted before being transferred into the cloud storage to enhance safety of the file. Equally, the symmetric key for encryption is encrypted with the use of the recipient's public key in the PGP encryption. It also explain cloud services ec2, cloud9 and throughput as in metrics.

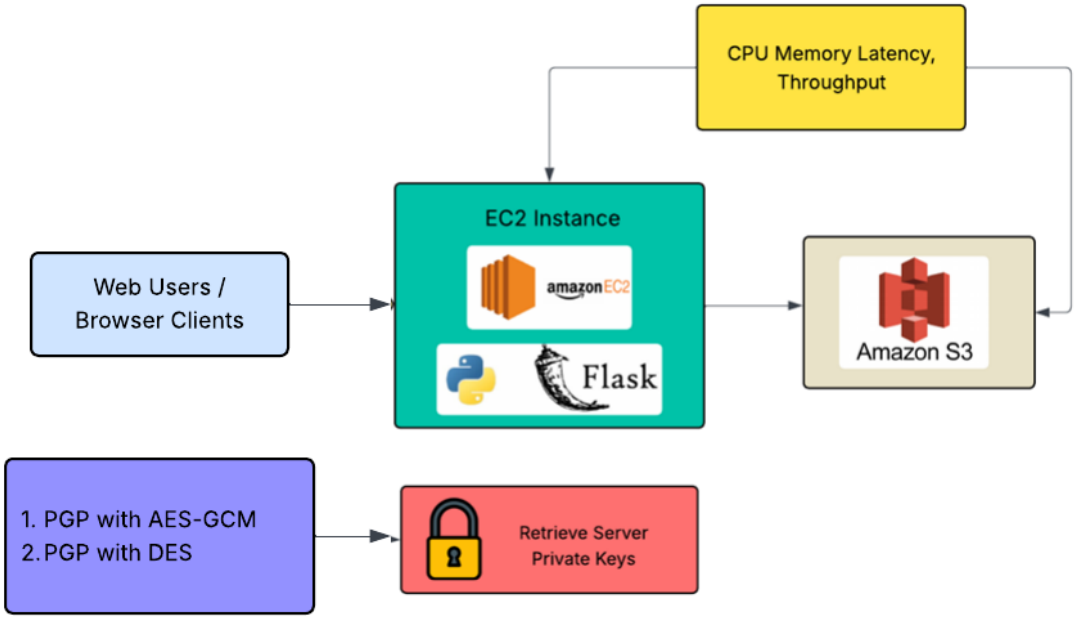

Figure 2: System Architecture Diagram

4.2 Algorithm 1: PGP-Inspired Hybrid Encryption Using ECDH and AES-GCM / DES

Algorithm 1 Hybrid Encryption Using ECDH with AES-GCM or DES

Input: Plaintext message M, Encryption mode $Mode \in \{\text{AES-GCM}, \text{DES}\}$
Output: Ciphertext C, Encrypted session key K_{enc}, Execution metrics T
// Key Generation Phase
Generate receiver key pair:
    $R_{priv}, R_{pub} \leftarrow$ X25519.generate()
Generate sender ephemeral key pair:
    $S_{priv}, S_{pub} \leftarrow$ X25519.generate()
// Shared Secret Computation
Compute shared secret:
    $SS \leftarrow \text{ECDH}(S_{priv}, R_{pub})$
// Key Derivation and Encryption
if $Mode ==$ AES-GCM **then**
  Derive symmetric key:
    $K \leftarrow \text{HKDF}(SS, 32 \text{ bytes})$
  Encrypt message:
    $C \leftarrow \text{AES\_GCM\_Encrypt}(M, K)$
else
  Derive symmetric key:
    $K \leftarrow \text{HKDF}(SS, 8 \text{ bytes})$
  Encrypt message:
    $C \leftarrow \text{DES\_CBC\_Encrypt}(M, K)$
end if
// Key Encapsulation
Compute encrypted session key:
    $K_{enc} \leftarrow \text{SHA256}(SS)$
// Performance Evaluation
Record encryption time and system metrics in T
Return C, K_{enc}, T

4.3 Justification of the Proposed Technique

The PGP-inspired hybrid encryption method is proposed in order to reach a secure and efficient compromise of confidentiality, performance and scalability of cloud-based settings. X25519 (ECDH) asymmetric key exchange provides secure session key setup and forward secrecy and AES-GCM symmetric encryption offers high-throughput cloud workload data protection that is fast and authenticated. DES only remains there because of legacy comparison to point out performance-security trade-offs. The design is stateless which enables cloud-native deployment and simultaneous access. This hybrid method is a way of combining the drawbacks of either symmetrical or asymmetric systems to achieve high security and low computational cost.

4.4 Ethical Considerations and Data Usage

In this study some ethical research principles have been followed because no real user data or personal information or sensitive records are captured, stored or manipulated during the system design or experimentation. Encryption and decryption are carried out on some synthetically generated plain text messages that are generated only to carry out the experimentation and performance estimation. No human subjects are included in the cloud deployment and load testing activities, and identifiable information is not used, which means that there are no risks of violating privacy or abusing data. Moreover, the implementation of the proposed system occurs in a controlled environment, in accordance with responsible security and ensuring that the system does not violate the ethical standards of academic research and institutional data protection policies.

5 Implementation

This chapter provides information on the practical implementation of the given encryption framework, tools, technologies, and cryptographic workflows. It also described the deployment of cloud through the AWS Cloud9, EC2, and Elastic Beanstalk in order to provide the support of a scalable and performance-monitored execution.

5.1 Tools and Technologies

Table 2 summarises the tools and techniques used for implementing, deploying and evaluating the proposed PGP-inspired hybrid encryption framework.

Table 2: Tools and Techniques Used in Implementation

Category	**Tool / Technique**	**Purpose in the System**
Programming Language	Python	Core implementation of cryptographic logic and web application
Web Framework	Flask	Development of stateless web-based encryption service
Asymmetric Cryptography	X25519 (ECDH)	Secure session key exchange with forward secrecy
Symmetric Cryptography	AES-GCM	Authenticated encryption ensuring confidentiality and integrity
Legacy Encryption	DES-CBC	Performance and security comparison with modern encryption
Key Derivation	HKDF	Secure derivation of symmetric session keys from shared secret
Hash Function	SHA-256	Encrypted session key representation (PGP simulation)
System Monitoring	psutil	Collection of CPU, memory, and throughput metrics
Load Testing	Locust	Simulation of concurrent users and scalability evaluation
Cloud IDE	AWS Cloud9	Development and testing environment
Cloud Deployment	AWS EC2	Hosting and execution of encryption services
Application Management	AWS Elastic Beanstalk	Deployment, scaling, and monitoring of Flask application

5.2 Encrypted Session Key Representation

The encrypted session key representation to imitate the PGP behavior is specified by hashing the shared secret that is produced during the ECDH key exchange with the hash algorithm named SHA-256. The result of the hash function of SHA-256 is regarded as a ciphered session key block and is saved as such. The process is the way in which the symmetric session keys are encapsulated by the PGP with the help of the public-key cryptography mechanisms. The system hides the key encapsulation step typically in a PGP-based encryption process by using a hash of the shared secret rather than simply storing it. Such representation does not divulge the crude shared secret and enables the symmetric encryption stage to run with safely gained key content. The encrypted session key blob is then utilized in the decryption process to recreate the symmetric session key, thus providing consistency between the encryption and the decryption processes without loss of consistency between the encryption and decryption stages and providing cryptographic isolation at the session level.

5.3 Encryption Workflow Implementation

Encryption workflow It starts with a plaintext message being provided to the system interface to be encrypted. A receiver key pair is then generated for each request and an ephemeral sender key pair is then generated. With these keys, a common secret is created by the ECDH key exchange mechanism. The secret shared is then converted into a symmetric session key by the HKDF process which depends on the chosen mode of encryption. Plaintext encryption may be performed in either AES-GCM or DES-CBC depending on the selected algorithm. In encryption, proper application of initialization vectors is used depending on the algorithm selected. The encrypted session key representation is produced and the ciphertext generated. During the process of encryption, the key generation time, key exchange time and symmetric encryption time are documented. Such a workflow provides a secure level of encryption and allows measuring performance metrics accurately to analyze them.

5.4 Decryption Workflow Implementation

The receiver then recomputes the shared secret with their own private key and the public key of the sender in the same ECDH process that it did in the encryption process in the decryption phase. The new calculated shared secret is then subjected to the same HKDF process to generate the symmetric session key and this makes it consistent with the encryption step. The ciphertext is decrypted using the same symmetric algorithm that was used during the encryption process to recover the original plaintext. The time taken to decrypt is measured to assess the computation cost that is incurred during the decryption process. It is a correct encryption and decryption process when the original plaintext is recovered successfully. This authentication makes sure that the key exchange, key derivation and symmetric decryption operation are properly synchronized so that the integrity and reliability of the proposed hybrid PGP- like encrypted systems are not compromised.

5.5 Implementation of AWS Deployment

5.5.1 AWS Cloud9 Deployment

The staging environment that is used to deploy the hybrid PGP encryption framework is AWS Cloud9. It offers an integrated development environment based on clouds which develops and executes the Flask application together with cryptographic modules and performance measurement logic. Cloud9 provides direct access to resources on AWS, to allow smooth integration with EC2 and Elastic Beanstalk in the development. The encryption procedures such as key generation, key exchange through ECDH, symmetric encryption and metric collection are verified in Cloud9 and then deployed. Cloud9 is used to achieve the consistency of the development and production environment and simplifies the process of dependency management and secure accessibility to the AWS infrastructure.

5.5.2 Amazon EC2 Deployment

The underlying compute infrastructure is Amazon EC2 on which the encryption application will be hosted. It offers scalable virtual machines needed to perform cryptographic

operations, process encryption and decryption requests, and to cater to the workloads of users concurrently created during load testing. The EC2 instances operate the Flask-based application and allocate CPU and memory resources to carry out cryptographic operations needed, including but not limited to AES-GCM, DES and ECDH key exchange. Using EC2, there is the possibility of assigning the computing resources in a controlled manner where one gets the opportunity to observe the CPU utilization, memory consumption, and throughput at different workloads. This system can be used to support the appropriate performance testing and scalability testing of the proposed encryption system. Flask web application is deployed and executed on AWS Elastic Beanstalk on a production-ready environment. Cloud9 application code is deployed on elastic beanstalk where the required EC2 instances are automatically created, application environment is configured and application scaling is monitored.

6 Evaluation

The proposed system was tested in this chapter based on performance, security and scalability measures like execution time, throughput, resource utilization, avalanche effect and load testing. The findings showed that PGP using AES-GCM is superior to the use of DES and the use of the baseline method, which prove the usefulness of the suggested cloud-native and forward-secure encryption system.

6.1 Evaluation Metrics and Performance Measures

Encryption Time (ms) refers to the time required to transform plaintext into ciphertext using the selected encryption algorithm.

$$\text{T\_enc=t\_cipher-t\_plain}$$

Decryption Time (ms) measures the time required to recover plaintext from ciphertext using the derived session key.

$$\text{T\_dec=t\_plain-t\_cipher}$$

Throughput (KB/s) shows the amount of data processed per unit time and reflects system performance under load.

$$\text{Throughput} = \frac{\text{Total data encrypted (KB)}}{\text{Total execution time (s)}} \times 100 \tag{1}$$

Avalanche Effect (%) evaluates cryptographic diffusion by measuring the percentage of output bit changes resulting from a single-bit change in input.

$$\text{Avalanche Score} = \frac{\text{Number of Flipped Bits}}{\text{Total Output Bits}} \times 100 \tag{2}$$

6.2 Experiment 1: Encryption and Decryption Time

Experiment 1 evaluates the performance of the hybrid encryption methods with the help of timing the encryption and decryption operations. PGP with AES-GCM shows lower encryption and decryption times as compared to PGP with DES, showing its performance for real-time cloud operations. The AES-GCM mode completes encryption in approximately 0.019 ms and decryption in 0.009 ms, whereas DES requires 0.086 ms and 0.043 ms, respectively as shown in Table 3. This experiment shows that AES-GCM is better suited for high-performance applications due to faster processing while DES, though functional, introduces higher computational overhead by making it less optimal for modern secure cloud environments.

Table 3: Encryption and Decryption Time for Hybrid Techniques

Techniques	**Encryption Time (ms)**	**Decryption Time (ms)**
PGP + AES-GCM	0.019	0.009
PGP + DES	0.086	0.043

6.3 Experiment 2: Throughput, CPU and Memory

Experiment 2 evaluates system performance by analyzing throughput, CPU usage and memory consumption for the two encryption techniques under typical workloads. PGP with AES-GCM achieves slightly lower throughput as compared to DES but shows higher CPU performance (0.9% vs 0.5%) and slightly lower memory usage as shown in Table 4. These results shows that AES-GCM provides a balanced performance with strong security and lower resource consumption by making it more suitable for scalable cloud deployment. DES while giving higher throughput, consumes more memory and processing time which reduce its overall performance in real-time secure environments.

Table 4: Performance Resource Utilization Comparison of Encryption Techniques

Techniques	**Throughput**	**CPU Usage (%)**	**Memory Usage (MB)**
PGP + AES-GCM	91.19	0.9	41.79
PGP + DES	111.77	0.5	43.99

6.4 Experiment 3: Avalanche Score

Figure 3 and Table 5 present a comparative analysis of the avalanche effect which has been achieved by PGP combined with AES-GCM and PGP combined with DES under different byte-level changes. The results shows that both encryption schemes having a strong avalanche effect with scores consistently above 93% having high sensitivity to input variations. However, PGP + AES-GCM outperforms PGP + DES in most cases by achieving higher avalanche scores which is particularly for byte changes 2, 4, and 5. This shows the superior diffusion capability of AES-GCM, making it more powerful and secure compared to DES in PGP-based encryption systems.

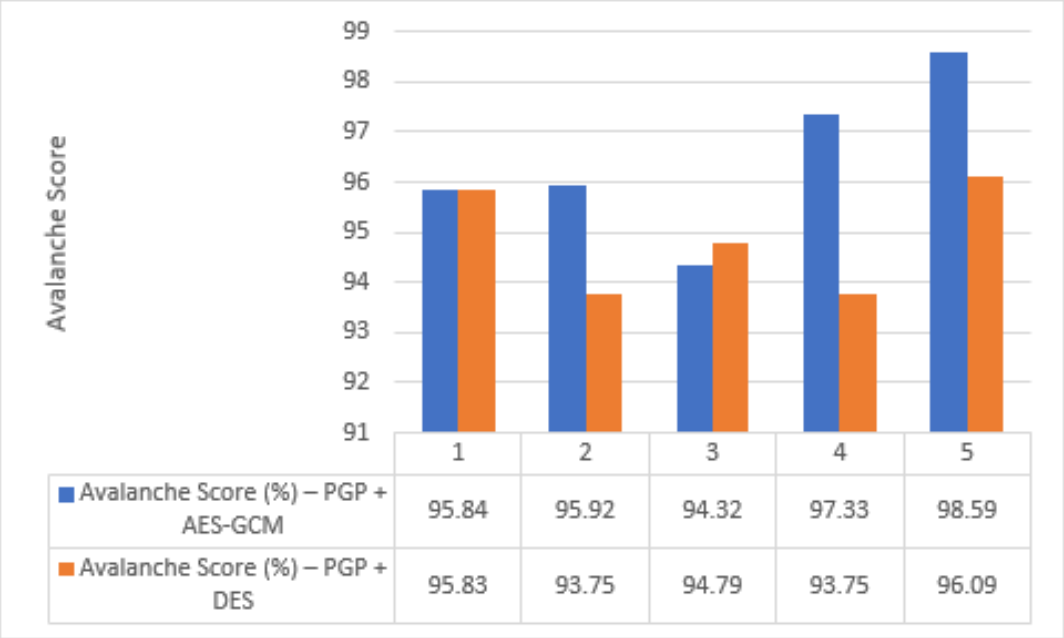

Figure 3: Avalanche Score Comparison using Byte-Level Changes

Table 5: Avalanche Effect Comparison for PGP + AES-GCM and PGP + DES

Change in Byte	**Avalanche Score (%) – PGP + AES-GCM**	**Avalanche Score (%) – PGP + DES**
1	95.84	95.83
2	95.92	93.75
3	94.32	94.79
4	97.33	93.75
5	98.59	96.09

6.5 Experiment 4: Locust Test with PGP+AES-GCM

The PGP was tested in AES-GCM encryption mode with a simulated concurrent workload in this experiment with the help of Locust. The test was conducted by using various POST requests to the /encrypt endpoint and monitoring the system behaviour when real-time encryption requests were made. Scalability and robustness were measured using metrics like request handling, throughput and stability. AES-GCM was always able to respond with low average response times of about 8 milliseconds and successfully attended to all requests without failure as shown in Table 6. This illustrates the ability of the algorithm to give secure and authenticated encryption, as well as maintain high performance with load. The findings show that AES-GCM is a good compromise between security and efficiency and can be used in the current cloud infrastructure.

Table 6: Load Testing Results for PGP with AES-GCM Encryption Mode

Parameter	**Value**
Total Requests	14,968
Failed Requests	0
Average Response Time (ms)	8.06
Minimum Response Time (ms)	4
Maximum Response Time (ms)	1,078
Average Payload Size (bytes)	10,579.05
Requests Per Second (RPS)	49.9
Failures Per Second	0

6.6 Experiment 5: Locust Test with PGP+DES

The results of this experiment have been used to test the PGP with DES encryption mode and Locust-based concurrent workloads. At the /encrypt endpoint, DES was serving several encryption requests and metrics like throughput and latency were measured as well as system stability. There were slightly lower average response time of the system with about 7.98 milliseconds but DES has been known to have known security weaknesses against AES-GCM as shown in Table 7. Although it did not crash on any requests, the lower cryptographic power renders it undesirable in the modern cloud applications. This experiment points out that despite the fact that DES can attain a higher throughput slightly, the AES-GCM is a better trade-off between the strong security assurance and the ability to perform well under high-load situations.

Table 7: Load Testing Results for PGP with DES Encryption Mode

Parameter	**Value**
Total Requests	13,833
Failed Requests	0
Average Response Time (ms)	7.98
Minimum Response Time (ms)	4
Maximum Response Time (ms)	450
Average Payload Size (bytes)	10,544.13
Requests Per Second (RPS)	49.41
Failures Per Second	0

6.7 Comparative Discussion with Baseline

The baseline study given by Shen (2023) proposes a hybrid AES-RSA encryption model mainly aimed at minimizing the execution time and maximizing the throughput based on Java-based execution model and offline experimentation on fixed data sizes. Whereas this study further develops the idea of hybrid encryption by adopting a PGP-inspired design using X25519 ECDH to exchange ephemeral keys and AES-GCM to do authenticated

encryption, implemented as a stateless web application in the cloud on AWS as shown in Table 8. In contrast to the baseline by Shen (2023) the work does not only test real-time encryption with workloads in parallel with Locust, but also analyzes specific metrics at the system level (CPU and memory use) and provides forward secrecy. An additional strength to the empirical evaluation is the use of DES solely to conduct controlled legacy comparison.

Table 8: Comparison Table of Baseline Paper vs Proposed Study

Aspect	**Baseline Paper Shen (2023)**	**Proposed Study**
Hybrid Design	AES + RSA	PGP-inspired X25519 + AES-GCM
Key Exchange	RSA static key encryption	Ephemeral ECDH with forward secrecy
Encryption Mode	AES (non-authenticated)	AES-GCM (authenticated encryption)
Deployment Environment	Local Java (Eclipse)	Cloud-native AWS (EC2, Elastic Beanstalk)
Workload Type	Offline, fixed-size data	Concurrent, real-time web workloads
Performance Metrics	Execution time, throughput	Time, CPU, memory, throughput
Load Testing	Not performed	Locust-based concurrent testing
Legacy Algorithm Comparison	Blowfish	DES (controlled comparison)
Security Properties	Confidentiality-focused	Confidentiality, integrity, authentication
System Architecture	Stateful, non-web	Stateless Flask-based web application

7 Conclusion and Future Work

This study has successfully designed, implemented, and tested a PGP-inspired hybrid encryption system with cloud-based application, which answered the research question on the role that hybrid key exchange using AES-GCM and DES has on security and performance when using cloud workloads simultaneously. The experimental findings indicate that the suggested framework successfully provides secure session key set up, forward secrecy, and stateless clouds when the ECDH with X25519 and HKDF based session keys are applied. Results indicate that PGP with AES-GCM is always better than PGP with DES in encryption and decryption latency, resource usage, and general compatibility with current cloud settings, secure able encryption, and better cryptographic assurances. Load testing with Locust was used to verify that the system can retain high concurrency with zero request failure and minimal response times. Even though DES had slightly better throughput, the security vulnerabilities of its use are well-known and make it inappropriate to use in practice, which supports AES-GCM as the best option. The research

has a limitation of the application of simulated workloads, limited choice of algorithm, and the lack of sophisticated key management services or hardware security modules. Future research can build upon this structure to include cloud-native, key management systems, analyze other modern algorithms, like ChaCha20-Poly1305, run in a multi-cloud implementation, and test against real-life production traffic to further prove scalability, resilience, and security in the real-life application.

References

Ajayi, R. (2025). Integrating iot and cloud computing for continuous process optimization in real time systems, *International Journal of Research Publication Reviews* **6**: 2540–2558.

Awadh, W. A., Hashim, M. S. and Alasady, A. S. (2024). Implementing the triple-data encryption standard for secure and efficient healthcare data storage in cloud computing environments, *Informatica* **48**.

Basco, J. Z. E., Romualdo, A. F., Saluta, J. A., Santiago, C. S. and Marfil, R. M. (2025). Comparison of data encryption standard (des) and advanced encryption standard (aes) in security issues of cloud computing: A literature review, *Innovations in Information and Decision Sciences*, Smart Innovation, Systems and Technologies, Springer Nature Singapore, pp. 189–200.

Bauskar, S. (2024). Advanced encryption techniques for enhancing data security in cloud computing environment, *SSRN Electronic Journal* .

Chatterjee, P., Bose, R., Banerjee, S. and Roy, S. (2023). Enhancing data security of cloud based lms, *Wireless Personal Communications* **130**: 1123–1139.

Chinnam, Y. and Sambana, B. (2024). Artificial intelligence enhanced security problems in real-time scenario using blowfish algorithm, *arXiv preprint* .

Dawood, M., Tu, S., Xiao, C., Alasmary, H., Waqas, M. and Rehman, S. U. (2023). Cyberattacks and security of cloud computing: A complete guideline, *Symmetry* **15**: 1981.

Elumalai, E. and Muruganandam, D. (2024). Secure and efficient data storage with rivest shamir adleman algorithm in cloud environment, *Bulletin of Electrical Engineering and Informatics* **13**: 2659–2667.

Jayalath, R. K., Ahmad, H., Goel, D., Syed, M. S. and Ullah, F. (2024). Microservice vulnerability analysis: A literature review with empirical insights, *IEEE Access* **12**: 155168–155204.

Kumar, D. D., Mukharzee, J. D., Reddy, C. V. D. and Rajagopal, S. M. (2024). Safe and secure communication using ssl/tls, *Proceedings of the International Conference on Emerging Smart Computing and Informatics (ESCI)*, IEEE, pp. 1–6.

Mahaboob Basha, S., Rishik, V., Naga Krishna, V. J. and Kavitha, S. (2023). Data security in cloud using advanced encryption standard, *International Conference on Inventive Computation Technologies (ICICT)*, IEEE, pp. 1108–1112.

Manjyanaik, H. N. B. and Mohanty, R. (2024). Preserving confidential data using improved rivest-shamir adleman to secure multi-cloud, *International Journal of Intelligent Engineering Systems* **17**: 162–171.

Patel, D., Patel, B., Vasa, J. and Patel, M. (2023). A comparison of the key size and security level of the ecc and rsa algorithms with a focus on cloud/fog computing, *ICT with Intelligent Applications*, Lecture Notes in Networks and Systems, Springer Nature Singapore, pp. 43–53.

Rakhra, M., Singh, A., Singh, D. et al. (2024). Digital signature verification in cloud computing, *2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, IEEE, pp. 1–6.

Revathy, J. S., Prakash, M. S., Lingeshwaran, K., Dhinakaran, J. D. and Harish, V. (2025). A comprehensive study of the advanced encryption standard (aes) for secure communications, *2025 3rd IEEE International Conference on Industrial Electronics: Developments & Applications (ICIDeA)*, IEEE, pp. 1–6.

Sarkar, B., Saha, A., Dutta, D., De Sarkar, G. and Karmakar, K. (2024). A survey on the advanced encryption standard (aes): a pillar of modern cryptography, *International Journal of Computer Science and Mobile Computing* **13**(4): 68–87.

Serengil, S. and Ozpinar, A. (2025). Lightdsa: A python-based hybrid digital signature library and performance analysis of rsa, dsa, ecdsa and eddsa, *arXiv preprint* .

Shen, D. (2023). Providing a hybrid and symmetric encryption solution to provide security in cloud data centers.

Shinde, R., Patil, S., Kotecha, K., Potdar, V., Selvachandran, G. and Abraham, A. (2024). Securing ai-based healthcare systems using blockchain technology: A state-of-the-art systematic literature review and future research directions, *Transactions on Emerging Telecommunications Technologies* **35**(1): e4884.

Singh, A. K. and Bhushan, K. (2025). In-depth literature review of cloud computing data hazards and mitigation strategies, *Concurrency and Computation: Practice and Experience* **37**(27-28): e70393.

Tajudeen, K. O., Ameen, A. O. and Adeniyi, A. E. (2025). A systematic review on advanced encryption standard cryptography to enhance message security, *Multimedia Tools and Applications* .

Vellela, S. S., Reddy, B. V., Chaitanya, K. K. and Rao, M. V. (2023). An integrated approach to improve e-healthcare system using dynamic cloud computing platform, *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, IEEE, pp. 776–782.