A Secure and Efficient Hybrid Cryptographic Framework for Mobile Cloud Systems Using ChaCha20 and Elliptic Curve Cryptography

MSc Research Project
MSc Cloud Computing

Vinay gunda
Student ID: x23302712

School of Computing
National College of Ireland

Supervisor: Shreyas Setlur Arun

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vinay gunda
Student ID:	X23302712
Programme:	MSC in Cloud Computing **Year:** 2024-2025
Module:	Research project
Supervisor:	Shreyas Setlur Arun
Submission Due Date:	11/08/2025
Project Title:	A Secure and Efficient Hybrid Cryptographic Framework For Mobile Cloud Systems Using ChaCha20 and Elliptic Curve Cryptography
Word Count:	……………………………… **Page Count**…20……………………………………….

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:	Vinay Gunda
Date:	11/8/2025

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):	

A Secure and Efficient Hybrid Cryptographic Framework for Mobile Cloud Systems Using ChaCha20 and Elliptic Curve Cryptography

Abstract

In the era of digital mobility, securing data transmission between mobile devices and cloud platforms has become a critical challenge. Cryptography plays a key role in ensuring confidentiality, integrity, and authenticity of sensitive information. Traditional models, such as RSA-AES combinations, often suffer from high computational overhead, increased latency, and inefficiency on resource-constrained devices, making them less suitable for real-time mobile cloud environments. To overcome these limitations, this study proposes a novel hybrid cryptographic framework combining Elliptic Curve Cryptography (ECC) for lightweight key exchange and ChaCha20, a stream cipher known for its speed and security, for data encryption. The system is implemented using an ASP.NET Core backend on Microsoft Azure, with secure storage in Azure SQL Database, offering a cloud-native architecture for encrypted data transmission and storage. The model is evaluated using JPG, PDF, and DOC files of varying sizes. Results show encryption times increase linearly with size, while decryption remains fast. The system achieved a peak throughput of 7.24 MB/s. These findings demonstrate that the ChaCha20-ECC hybrid model ensures high performance, scalability, and strong encryption, making it ideal for secure, real-time communication in modern mobile cloud applications.

Keywords: Hybrid Encryption, ChaCha20, Elliptic Curve Cryptography (ECC), Mobile Cloud Security, Secure Data Transmission.

1 Introduction

1.1 Background

Cloud computing has revolutionized the way data is processed, stored, and accessed, offering scalable, on-demand services to individuals and enterprises. With the proliferation of mobile technologies, cloud integration into mobile platforms has led to what is now known as mobile cloud computing (MCC) by (Muheidat and Tawalbeh, 2021). This paradigm enables resource-constrained mobile devices to offload computation and storage tasks to powerful cloud infrastructures, significantly enhancing performance and usability. However, this integration also opens up critical challenges in terms of data security, privacy, and transmission efficiency.

Data exchanged between mobile devices and cloud servers is often sensitive—ranging from personal user information to corporate data via cloud as shown in Figure 1. This makes it a prime target for cyberattacks such as eavesdropping, data breaches, and man-in-the-middle attacks. Furthermore, traditional encryption algorithms, although secure, typically impose high computational and energy overheads, making them unsuitable for mobile devices with limited processing power and battery life.

To address these issues, lightweight and efficient encryption mechanisms tailored for MCC environments are essential. Hybrid encryption, which combines the speed of symmetric algorithms with the robust key management of asymmetric methods, has emerged as a promising solution. This study focuses on optimizing such a hybrid encryption model using ChaCha20, a lightweight and fast symmetric cipher, alongside Elliptic Curve Cryptography (ECC), known for its strength in public key operations with smaller key sizes.

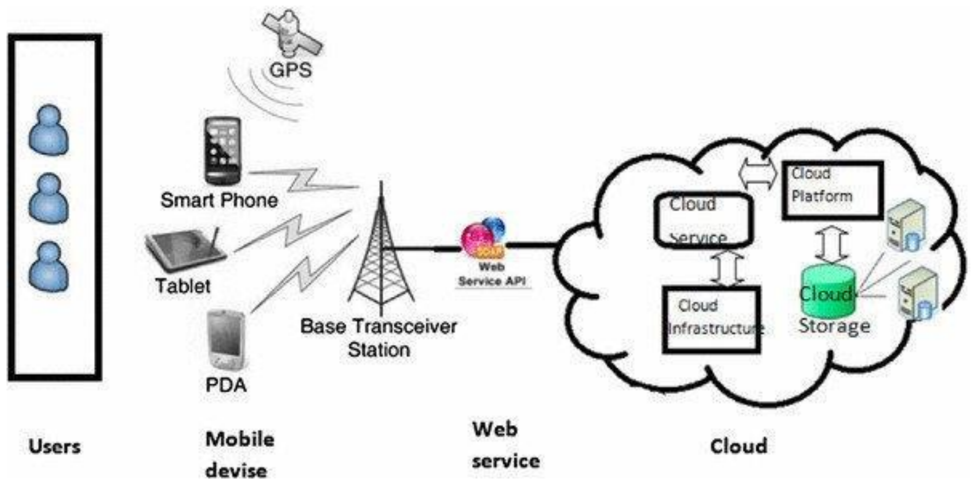

Figure 1: Architecture of Mobile Cloud Computing by Singh et al. (2019)

1.2 Aim

The main aim of the proposed study is to develop and realize a new hybrid strategy encryption on the basis of a combination of ChaCha20 and ECC to improve data security and efficiency in the mobile cloud computing scenario, especially in resource scarce settings. As the use of cloud services gains prominence on mobile platforms, there has been the proliferation of vulnerability in regards to privacy of data that needs to be transferred and stored. Although they are strong, traditional encryption mechanisms require too much computing that cannot be accommodated in devices with low computational power and battery power. This study aims at overcoming these drawbacks, by combining the lightweight symmetric encryption of ChaCha20 with the high power of asymmetric key exchange in ECC. On a cloud side, the offered solution guarantees confidentiality, integrity, and minimal latency without seriously loading the client device or cloud infrastructure. The objective is to allow mobile communication to cloud servers to be seamless and safe and also optimising the resources used. Also the hybrid scheme is expected to cut down on the complexity of key management, and enhance scalability in distributed cloud systems. The study is aimed at displaying an enhanced level of the performance indicators like the speed of encryption performance, throughput, and power efficiency by conducting an experiment with the identified model in numerous situations best suited in real-time cloud-based environments thus introducing more practicable and more secure mobile cloud services.

1.3 Research Question

What impact does the proposed ChaCha20-ECC hybrid cryptographic model have on encryption efficiency and data security in mobile cloud environments?

1.4 Objectives of the Research

The research objectives for this report are:

1. To design and implement a novel hybrid encryption model that combines ChaCha20 and ECC to enhance data confidentiality, integrity, and transmission efficiency in mobile cloud computing.
2. To evaluate the performance of the proposed ChaCha20-ECC hybrid scheme in terms of computational speed, energy consumption, and latency on resource-constrained mobile devices compared to conventional encryption techniques.
3. To assess the scalability and key management efficiency of the hybrid encryption framework in distributed mobile cloud environments, ensuring secure communication with minimal overhead.

1.5 Outline of the Report

This report is structured into the following sections:

1. Chapter 1: Introduction: Introduces the need for secure mobile cloud communication and highlights the challenges in traditional cryptographic approaches.
2. Chapter 2: Literature Review: Reviews existing encryption models and identifies research gaps in latency, key exchange, and performance.
3. Chapter 3: System Design: Describes the architecture of the proposed ChaCha20-ECC hybrid model and its integration with Microsoft Azure services.
4. Chapter 4: Methodology: Details the implementation approach, cryptographic algorithms, encryption/decryption workflow, and cloud environment setup.
5. Chapter 5: Implementation: Presents the mobile app, backend services, and secure cloud storage components developed using ASP.NET and Azure SQL.
6. Chapter 6: Evaluation: Analyzes encryption/decryption performance across file types and sizes, including calculated latency and throughput.

7. Chapter 7: Conclusion and Future Work: Summarizes the research contributions, acknowledges limitations, and suggests future enhancements to the hybrid model.

2 Related Work

2.1 Evolution of Cloud Computing and Its Security Landscape

Cloud computing has become one of the game-changing paradigms of the digital era, which has changed the way people and organizations store, organize, and retrieve data and applications Umar and Rana (2024). Since its inception offering simplest storage and computing facilities, cloud computing has since developed into a very complex infrastructure including Infrastructure- as-a-Service (IaaS) to Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS), easily scalable resources at internet-enabled locations. This advancement has given businesses the ability to optimize processes, decrease the expenses on hardware, and innovation. Nevertheless, as the speed of adoption and growth of cloud services has been fast we have also experienced considerable problems especially in the sphere of cybersecurity (Alkadi et al., 2020). When data is clouded, it is exposed to unauthorized users and since data goes on the cloud, there is a chance of data breach and even hacking and cyberattacks. Security of the cloud computing environment has therefore gone hand in hand with this transformation shifting towards the use of encryption, zero-trust and AI-based threat detection approaches. The modern cloud security models focus on data confidentiality, integrity and availability particularly when dealing with shared responsibility models of cloud providers and users. It also shows a change in the security stance to compliance-based security, wherein companies are required to achieve high standards like GDPR, HIPAA, and ISO 27001. In addition, loss of control over the security layer resulting in proliferation of mobile and edge devices connected to the cloud has increased the threat surface contributing to the need of implementing more stable and flexible security layers. Hybrid cryptography Mixing the advantages of symmetric and asymmetric cryptography system Hybrid cryptography has received attention due to its capability to deliver quick speed as well as sending out secure key. These strategies are getting more vital especially in resource-conscious and distributed environment nowadays where performance should always be maintained despite having a robust security. With cloud computing playing a central role in most of the current digital environments, it is vital to continually transform its security policies to prevent unauthorized access to confidential information or lack of faith to cloud-based environments.

2.2 Role of Cloud in Handling Resource Constraints

Due to the resource limitations of the environment (including mobile devices, edge networks, and developing areas with little computing capacity or unstable connectivity), cloud computing is at the center of a balance between need and availability (Kai et al., 2021). Mobile devices (and low-power ones too) can utilize the resources of high-performance computing without actually having to integrate them into the device design due to offloading the storage and processing to discreet cloud-based servers. This feature supports high-performance apps like artificial intelligence, live analytics, and Big Data processing on low-end hardware effectively. Moreover, cloud services provide elasticity giving its users the ability to expand or decrease resources depending on the workload needs hence creating the optimal allocation of resources and minimizing overheard (Huang et al., 2021). It is a valuable model in the mobile cloud computing (MCC) environment where devices mostly rely on obscure servers to perform complicated jobs to warrant user experience and applicative reaction. Also, lightweight deployment models such as microservices and containerization are supported by the cloud that reduces the local resource utilisation further. Yet, even as cloud computing solves some limitation problems in computations, it presents other security and privacy issues especially concerning the transmission and storage of data across public networks. In these scenarios, we need to resort to the use of encryption mechanisms that should not only be secure, but also light and power-saving. The use of hybrid encryption scheme which can be adopted to mobile and limited systems is also to ensure that the sensitive data is secured and that there is no overloading of the battery or the processing capacity of the device. Besides, cloud service providers are also providing enhanced mobile-specific APIs, resource scheduling, and optimized communication protocols that are more focused on achieving high performance and at the same time conserving plenty of energy. In short, cloud computing plays a major supporting role to address the shortage of resources to deliver rich functions on the thin-client and achieving stable balance between efficiency, cost, and security.

2.3 Cryptographic Techniques in Mobile Cloud Computing

(Shabbir et al., 2021) developed a model of requirement-oriented health information security related to Mobile Cloud Computing (MCC) that should be deployed in healthcare and its proposed implementation of the Modular Encryption Standard (MES), a layered approach to security development in addressing the severe needs of privacy and safety of dealing with sensitive data in health care. The need to have a study was because of the urgent need of effective data protection in the MCC environments with the issues related with the exposure of health information to vulnerabilities and breaches becoming the stain of the full-scale implementation of cloud-based solutions. To do this, an MES algorithm has been presented and compared to other known cryptographic algorithms including but not limited to AES, blowfish, RC5, RC6, DES, and 3DES. To conduct the comparative analysis of the performance, the diagnostic tools of Visual Studio were utilized as the research team paid special attention to memory usage as one of the most critical measurements. The results demonstrated that MES could easily beat traditional algorithms, where the execution time (10.043 seconds) was extremely low and memory consumption was quite low representing efficiency in MCC environments due to resources limitation. The approach, however, does not have limits. The research predominantly focused on the memory utilization but did not focus on other important parameters like encryption/decryption speed, energy efficiency, scalability in working with numerous different data formats across the healthcare industry, or rather resisting advanced cyber-attacks. In addition, the real-life application and implementation of MES on different healthcare systems within MCC is yet to be tested. In spite of these constraints, the study is valuable, as it involves the novel introduction of the targeted and modular type of encryption scheme that can enhance the performances of mobile cloud healthcare system and increase qualitative security assurance of this system.

In (Maria Navin and M Lutimath, 2023.), the authors have contributed to the improved encryption of the data altogether which will be risk-free and highly productive to implement in a resource constrained mobile cloud environment by boosting the basic Advanced Encryption Standard (AES) algorithm. In the research, the researchers tried to answer the growing concerns regarding the data privacy and latency with the cloud-based systems, especially the client devices with low computational capacity. The paper suggested the following implementation method: to optimize the AES operations namely shift rows and mix columns operation which contributes to AES execution latency up to 60 percent. Besides, the system combines Elliptic Curve Cryptography (ECC) protocol with the altered AES to further strengthen data integrity, as well as authentication. The rationale behind developing this hybrid cryptographic method is to offer low-power, low-latency symmetric encryption solution of mobile cloud computing. Testing and Evaluation indicated that the suggested system was proven to be more successful and safer than the techniques currently available, which means that this is a viable method in regard to securing cloud data transfer. The limitation of the study is, however, that these computational enhancements are being focused on rather than the evaluation in terms of energy usage, resistant to advanced cryptographic attacks and application in real life situations in heterogeneous mobile networks. Nevertheless, the contributions of the work towards such optimisation of an encryption process in AES and the support of such optimisation by ECC cannot be underestimated as it further protects mobile cloud data.

Recently, the authors of (Awan et al., 2020) suggested an augmented framework of encryption with the purpose of enhancing protection, performance, and efficiency of cloud-assisted mobile devices by modifying a standard AES-128 algorithm. The essence of the project lied in the fact that, the study was aimed at solving the resultant security issues, and resource limits in the computational clouds and to a great extent in outsourcing its sensitive data to smart mobile devices. The suggested plan will present a dual round key model instead of the usual single round key, and the speed of encryption will go up to 1000 blocks per second as compared with 800 blocks per second. This change is not only performance increasing but also has a positive effect on power consumption, increased load balancing and increased trust and resource management in the context of the whole network. The simulation based analysis required the enhanced AES to be implemented against different plaintext sizes (16, 32, 64, and 128 bytes) and the result showed great improvements such as; the project led to the reduction of the energy requirements by 14.43 percent, the network requirements by 11.53 percent, and the delay by 15.67 percent. These results indicate that the framework is very appropriate to be deployed safely and efficiently in cloud services. The limitations of the study are however that the algorithm is not tested in the real environment and the potential capability of the cyber-attacks or side-channels exploiting the algorithm were not examined. Also, it has not been tested on various network experiences and the possibility of scaling to various cloud systems. Nevertheless, the study offers an interesting improvement resting on the AES to resource-effective and safe mobile cloud technologies.

Another study by (Adeniyi et al., 2023) suggests an alternative to the AES algorithm that would increase the security and efficiency of securing confidential medical data of patients, given that even minor changes in this information could result in misdiagnosis and lifelong consequences. Since the transition to the electronic health records (EHRs), cyberattacks are more of a risk, which entails powerful encryption. The proposed study was to minimize the computation budget of conventional AES and make it more effective in the protection of healthcare information. The suggested solution consisted of the alteration of the last stage of AES algorithm and its testing with scrambled input sets of different size and type of content. It was observed that the modified AES encrypted faster (average of 1293.837 ms vs. 1513.3 ms in the conventional AES), with the conventional one performing significantly faster during a decryption phase (1289.627 ms vs. 1400.136 ms of the modified AES). Also, the Avalanche effect proved that modified AES gained better security in case of small files, and conventional AES worked better in terms of diffusion of large files. This means that the modified AES can be more applicable on light weight health data encryption and not very good on bulk data decryption. The most important constraint in the present study is its asymmetric performance, in which performance on encryption is maximized but performance on decryption is not, and this performance fluctuates with file size by a wide margin. In addition, the study did not conduct real-time experiments in a practical healthcare setting, which restricts the generalizability of this study. However, the study brings an intelligent improvement to the protection of small-sized EHRs at lower encryption overheads.

Two years later, the authors in (Ajagbe et al., 2024) suggested using a multi-level security where they adopted Huffman coding with symmetric encryption algorithms, especially Advanced Encryption Standard (AES), to increase security of Personally Identifying Information (PII) during transmission. The reason that this study was to be carried out was due to the fact that encoded bits and wordlists are susceptible to tampering, and they take too much time to be transmitted when they have to be sent individually. Another major issue was that there are no solid error-correcting mechanisms on a Huffman coded data and that entire information would be lost in case of corruption of a bit. The researchers carried out an experience under the simulated condition to achieve a simulation environment of the secure communication and to trace transmission interference through Mifare Classic 1k RFID cards. In Cipher Block Chaining (CBC) and in Electronic Codebook (ECB), AES was tested with a dataset size of N = 200 taking the execution time in both cases AES 0.023 ms at a unique character probability of 0.5. The research proved that, by altering bytes and by changing the number of characters that were unique, it was possible to effectively simulate the behavior of AES, meaning it was appropriate to use in cases of secure RFID communication. Weaknesses also come in the form of how the study is based on simulation and there has been no actual implementation into practice to determine how the system will handle the attack or how it will react to faulty bits. Moreover, though AES provides data confidentiality, there is always the possibility that Huffman coding lacks a built-in correction of errors and therefore, there might be loopholes in data reliability. The study has its limitations notwithstanding, the design provides the layered cryptography mechanism that is expected to minimise the effect of signal interference during communication and improve the data security over RFID.

Another study suggested a rather compact and safe mobile cloud computing model in (Kurunthachalam and Ahmed, 2025) by combining the Diffie-Hellman key exchange encryption algorithm with BLAKE2 hashing method to provide efficient and secure user identification. This study was meant to overcome the drawbacks of the contemporary security measures, which are either inefficient to be used with mobile devices or not secure enough to be applied to conduct sensitive data cross the open networks. The suggested solution will be to generate the user-accessible keys (public-private keys) using Diffie-Hellman at the moment of registration, and apply BLAKE2 hashing on the client side with the intent of non-differentially homomorphically enabling the users authentication process. The performance of the system was tested and measured by encrypting and decrypting times comprised 1140 ms and 1265 ms that are considerably lower than with conventional procedures and are still quite sufficient with a definite level of security equaling to 98.1%. The combination was superior in its efficiency, lower use of resources, and efficient processing of data in terms of time that would fit the notion of mobility application. But one of the limitations noted is that it has a slightly lower security margin, and a small set of attacks are resistant to other stronger encryption schemes and there is little analysis of the behavior of the system in the real world application and subject to deployment in large scale applications. Nevertheless, these disadvantages do not diminish the fact that the study successfully introduces a secure and efficient model of protection that matches the performance that is supposed to be relevant in mobile cloud systems.

(Prakash et al., 2024) developed a secure system of text communication by combining encryption and key exchange protocols to secure confidential information being conveyed on unsecured networks. This study aimed to accomplish confidentiality and the privacy of messages sent by the users by exploiting the limitations of the human perception and methods in cryptography that hide the presence of secret information. In the given approach, user text is encrypted with a shared key, which is securely exchanged with the help of the Diffie-Hellman key exchange process a method of public-key cryptography that enables secure generation of a common secret-key... This encryption message (ciphertext) is sent together with the recipient and it should not be decrypted without the right common key. The project information including all the possible data can be collected in a remote cloud MySQL database and be accessed via a cloud connection through the web. This system provides that even when the message is intercepted by a third party, he or she will enjoy the message without the encryption key. This experiment was able to show that it is viable to use cloud infrastructure in securing message exchange using keys. Nevertheless, its central drawback is an apparent reality of the need to rely on manual exchange of keys between the senders and receivers, which may prove to be a potential source of concern especially when transmitted via loose mediums. Also, the system can have a narrow scale as it involves text-based encryption and cannot scale up to other types of data or multi-user systems. With these limitations notwithstanding, the study has been able to demonstrate a very minimal but powerful encryption framework that can be used between the cloud, and the cloud where Diffie-Hellman is used to provide secure communication.

2.4 Security Protocols and Key Management in Cloud Systems

Security protocols, and key management systems are prerequisites in sustaining trust and data protection in cloud computing environments (Hazra et al., 2024). A common protocol to protect the communication between a client and a cloud server is TLS/SSL, IPsec and HTTPS, which protect data-in-transit by making interception and manipulation impossible or arguably difficult. In the meantime, data-at-rest typically employs some mechanism such as AES (Advanced Encryption Standard), or RSA (RivestShamir-Adleman) to maintain confidentiality. Nevertheless, the success of these security measures greatly rests on the strength of key management procedures. The most important key management activities involve the creation, sharing, storing, changes and withdrawals of cryptographic keys, which require security to avoid vulnerabilities that may lead to unauthorized usage or loss of information. Key management in cloud systems (especially in multi-tenant clouds) is a hard task due to the variety of keys to be managed between various consumers and contexts (Pallavi and Jayarekha, 2022). Providers such as the AWS, Azure, and Google cloud provide Cloud Key Management Services (KMS) to enable centralized control of key lifecycle, with the ability to integrate it with hardware security modules (HSMs), to provide added protection. But these centralized systems have the potential of becoming single points of failure when not well guarded. Moreover, hybrid and multi-cloud implementations require mechanical matchups and end-to-end key management which is sophisticated to run. Hybrid encryption has found itself to be a relatively easier way out of most of these problems as it allows to use both the speed and the symmetric encryption along with some of the secure key exchange characteristics of asymmetric encryption. This solution will be especially relevant in mobile cloud conditions when the performance and low latency counts. Weightless encryption algorithms, and distributed key exchange protocols are currently under investigation so as to accommodate the needs of constrained devices. Further, emerging paradigms like attribute-based encryption (ABE), and homomorphic encryption are also under exploration to provide fine-grained access control and computation over encrypted information.

3 Research Methodology

3.1 Elliptic Curve Cryptography (ECC)

The Elliptic Curve Cryptography (ECC) is an asymmetric cryptographic algorithm that aims to ensure a high degree of security with significantly smaller key lengths, which makes it quite well-suited to resource-limited computing contexts, such as those pertaining to the mobile cloud computing. ECC relies on the field of mathematics involving elliptic curves over finite fields and offers safe functions in key exchange, digital signals, and encryptions. To be implemented in the proposed framework, ECC is to be utilised specifically as a secure communication between mobile clients and cloud servers (namely Elliptic Curve Diffie-Hellman (ECDH) key exchange). Based on the ECC, a shared secret key is established between the two parties without them communicating the secret in the first place, namely, without using the secret messages to reduce the risk of being overheard and attacked by a man in the middle. ECC has the advantage of efficiency, in that the size of keys is shorter: an ECC of 256 bits achieves the same security as a cryptographic key of 3072 bits in RSA. These savings in key size are directly proportional to minimal and rapid computations and minimal use of energy, which is essential in mobile devices and their less battery durability and processing power. In terms of computing cost, the operations in the ECC are cheaper especially point multiplication when compared to other operations in the RSA since they include exponentiation of large integers. The ECC phase guarantees sharing of the common symmetric key by the sender and receiver without the exposure in a transfer and creates a firm background of the remaining cryptography procedure.

3.2 ChaCha20 Stream Cipher

ChaCha20 is a modern stream cipher that can offer symmetric encryption that is both fast and secure with low computational cost, which is suitable in a mobile and embedded application. ChaCha20 was developed by Daniel J. Bernstein as an enhancement of Salsa20 cipher aiming at resolving various performance and security issues involved in usage of earlier algorithms such as AES, particularly with regard to software implementations. In contrast with block ciphers, ChaCha20 works with 512-bit blocks, has a 256-bit key and a 96-bit nonce to produce pseudo-random keystream which is bitwise XORed with plaintext to form ciphertext. Invoked in this study, ChaCha20 is an algorithm to cipher data payloads once a trusted session key is generated with ECC. It has the strongest points of being very fast, with strong encryption and decryption, its minimal set-up cost, resistance to side-channels (especially timing attacks), and that it is extremely appropriate with devices of low computation power.

Where AES does not scale well on hardware-less devices, ChaCha20 continues to perform well in pure software, on devices like mobile phones and cloud containers.

This figure 2 shows the proposed workflow for a secure mobile-based file encryption and upload system, segmented into three layers: Mobile Client, Backend Services, and Cloud Storage. The Mobile Client layer handles file selection through an Android app, ECC (Elliptic Curve Cryptography) key generation and ChaCha20 encryption, all processed locally to ensure confidentiality. The Backend Services include a RESTful API developed in ASP.NET Core that facilitates secure communication, ECDH (Elliptic Curve Diffie-Hellman) key exchange to derive a shared secret, and Azure App Services for encrypted processing and coordination. The Cloud Storage layer comprises an Azure SQL Database for storing metadata, secure file storage for encrypted content, and security logs for maintaining audit trails. Below, the Encryption Process Flow outlines the sequence: ECC key pair generation initiates secure communication, followed by ECDH key exchange to produce a symmetric key. This key enables ChaCha20 encryption, after which a SHA-256 hash ensures data integrity before files are securely uploaded. This layered architecture ensures end-to-end encryption, secure transmission, and reliable storage, leveraging modern cryptographic techniques and Azure cloud services to ensure data privacy and integrity in mobile-cloud ecosystems.

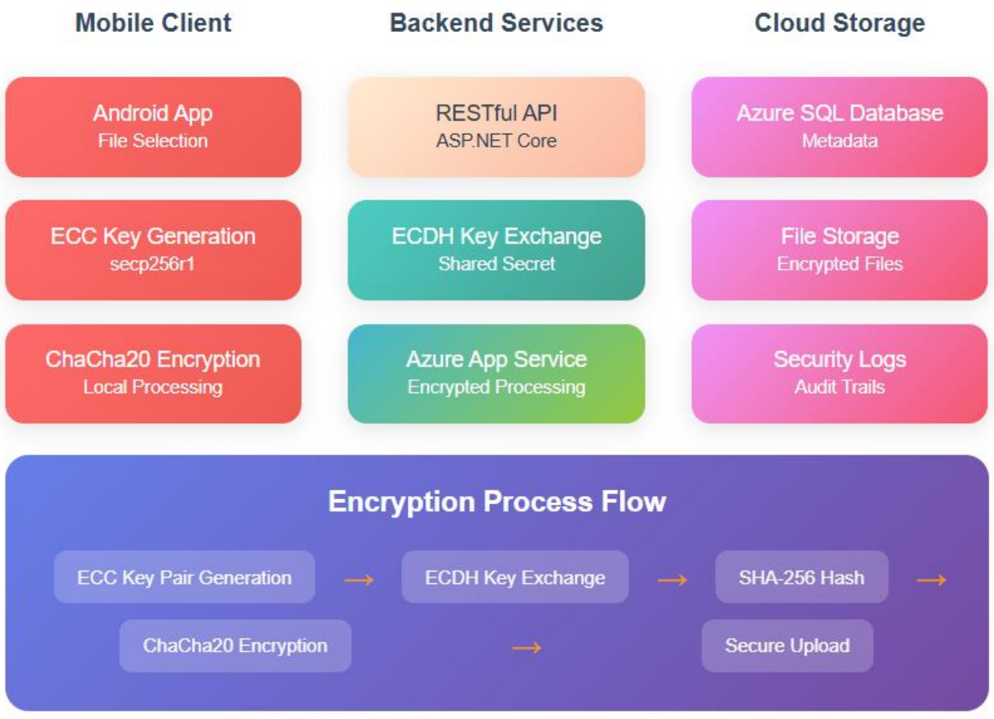

Figure 2: Proposed Workflow Diagram

4 Design Specification

Figure 3 shows the secure file encryption and decryption workflow in the proposed ChaCha20-ECC hybrid cryptographic framework. The process begins with user registration; if successful, the user’s details are stored in an Azure SQL Database. Returning users log in, and upon successful authentication, they access the home page. For encryption, when the user uploads a file, the system generates an ECC key pair, performs ChaCha20 encryption on the file, and applies a hash operation for integrity verification. These operations occur within an Azure App Service, ensuring secure and efficient processing. The encrypted file is then securely stored, and the encryption key is sent to the user. For decryption, the user requests to view/download a file, triggering the secret key expansion process. The ChaCha20 decryption algorithm is applied, followed by signature verification and hash validation to confirm file integrity. The decrypted file is then downloaded to the user's device. Throughout the process, Azure Monitor provides system performance and security tracking. This architecture ensures end-to-end encryption, secure key management, and robust integrity verification, enabling high-performance, low-latency secure file handling in mobile-cloud environments.

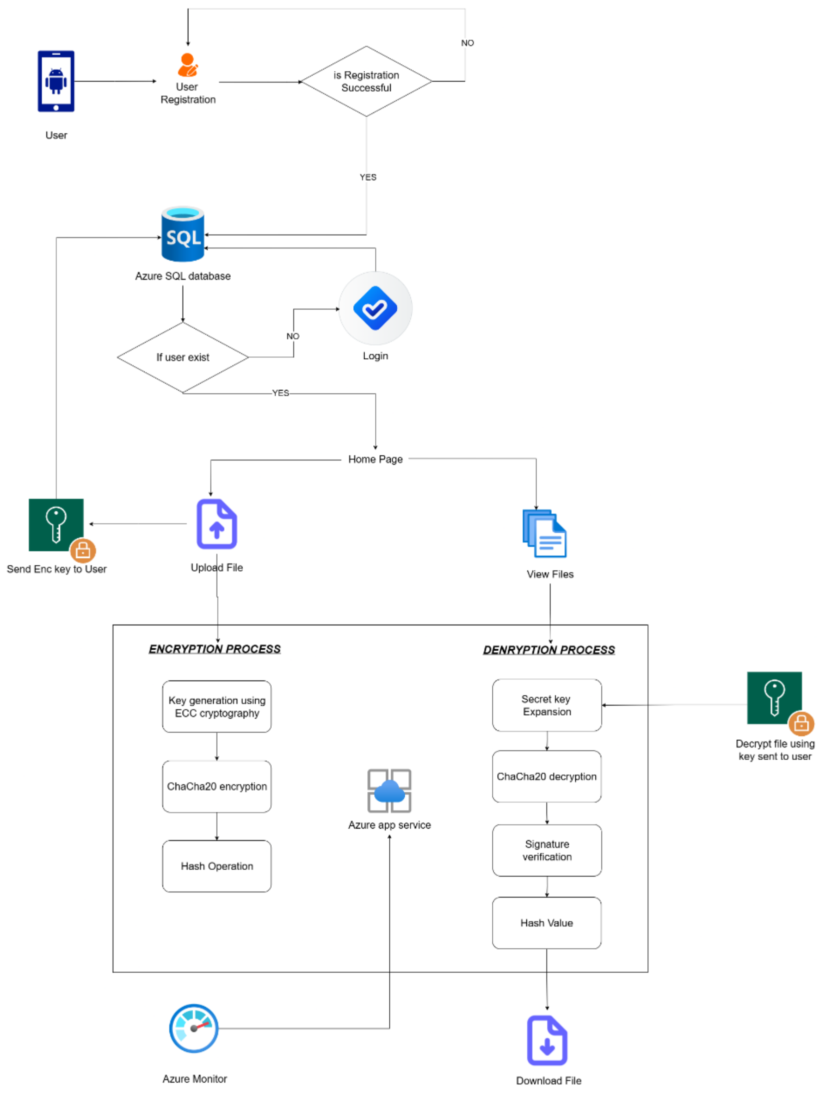

Figure 3: System Architecture Diagram

5 Implementation

5.1 Tools and Technologies

Table 1 shows tools and technologies for this study.

Table 1: Tools and Technologies

Tool/Technology	Purpose in ChaCha20 + ECC-Based Hybrid Cryptographic Framework
Android Studio	Development environment for building the mobile application that allows users to encrypt and upload data.
Java	Programming language used in the mobile app to manage ECC key generation and initiate ChaCha20 encryption.
Android OS	Platform for running the secure mobile application supporting cryptographic operations and RESTful communication.
Azure Cloud	Centralized platform used to manage encrypted data storage and integrate compute resources securely.
Azure App Service	Cloud-hosted environment for deploying the backend RESTful API that processes encrypted file uploads.
Azure SQL Database	Relational storage for managing metadata, key references, and transaction logs associated with encryption sessions.
Python / C#	Programming languages used to develop middleware and backend cryptographic modules (ChaCha20, ECC) and handle server logic.
RESTful API	Interface for secure communication between the mobile app and cloud services during file and key transfer.

5.2 Mobile Application Layer

The mobile layer of application creates the user interface with the data transmission security model proposed. This app that is created with the help of Android Studio and needs to be installed on Android operating system will provide the user with the ability to securely upload sensitive data safely on the cloud and encrypt it at the same time. The application is designed in such a way that a user is capable of opening files locally and initiating the encryption procedure on mobile devices. When a file has been selected, the app commences with the creation of the ECC (Elliptic Curve Cryptography) public-private keys. This is either implemented as a light weight Python backend (called using remote via REST API) or pre-created using a hybrid model included with the application. This public key is then transmitted to the ASP.NET Core backend server running at the Microsoft Azure and a secure, authenticated session key is derived. At the same time, the app does the setup of the ChaCha20 stream cipher given a 256-bit symmetric key (hashed from the ECC shared secret with sha-256) and a 96-bit nonce of random value. ChaCha20 encrypts the file completely on the machine so that no plaintext information is ever sent. The secure upload then in turn uploads the encrypted file, the nonce and metadata (timestamp, filename, hash) using HTTPS the ASP.NET server. Some other features are that there are indicators on file status, session management, and upload tracking which is encrypted. The mobile application reduces the risk exposure by running the encryption on the device and can be consistent with zero-trust cloud concepts. In addition, it uses ChaCha20, which makes even the lower-end mobile devices undergo rapid, low-power encryption. The mobile app connects with the cloud by means of a RESTful API, such that uploads are securely and in real-time. This will be the initial crux of your new hybrid encryption and decryption process using ChaCha20-ECC with the type of cloud integration i.e. the hybrid process of encryption and decryption of data that can operate at mobile level and that can also be interoperable with the cloud using Microsoft Azure.

5.3 Backend Cryptographic Service

The backend cryptographic service is implemented as a RESTful web API with the ASP.NET Core and it is hosted on Microsoft Azure App Services. It serves as the liaison between the mobile application and cloud-based storage as well as plays important cryptographic roles. On the reception of the encrypted file and the ECC public key of the client, the backend service will at first obtain its ECC key pair with the help of the .NET namespace System.Security.Cryptography. The Elliptic Curve Diffie-Hellman (ECDH) protocol can then be used to calculate a common secret using the obtained client published key. it is hashed with the SHA-256 hash algorithm in order to generate a symmetric 256-bit session key to be used in optional verification and auditing. A backend could also copy encryption of ChaCha20 with same nonce over to integrity check (necessary to run during implementation). After verification, the backend saves the encrypted file in a secured location on the Azure server or alternatively on the Azure Blob Storage in case of future scaling. The filename, the nonce, the timestamp of the encryption and the file checksum are filed in a Microsoft Azure SQL Database, a highly available and scalable relation data storage. Its database schema consists of user session tables, file logs, and encryption metadata tables and upload audit trails. ASP.NET Core backend has been secured against unauthorized access through token authentication based on OAuth2.0 and it is on HTTPS. It also connects to Azure Monitor and Application Insights to get real-time analytic and diagnostics and security warnings. This backend layer provides data secrecy and accountability of the system because it maintains end to end encryption. Serving it on the cloud ecosystem of Microsoft Windows, issues of seamless integration, automated scaling and strong security that documents are pertinent and matter in real time transmission of encrypted data and cloud retention in a mobile cloud infrastructure have been addressed.

5.4 Secure Cloud Storage and Azure SQL Database

In the suggested implementation, instead of object storage such as Amazon S3, encrypted files and related metadata are stored in Azure SQL Database or Azure App Services file system. When the encrypted file is received by the mobile client and optionally checked on the backend, they are kept safely in the directory on the server side, located behind ASP.NET Core on Azure App Service or provisioned Azure File Share, according to the deployment requirements. At the same time, metadata related to the filename, nonce, file checksum (SHA-256), upload timestamp, and encryption session key ID is not stored in the Azure SQL Database. Such relational storage means that above encrypted files and the metadata related to it can be indexed queried and integrity checked with high performance. Every SQL database record refers to the physical file path and is associated with the ECC key pair that will be employed to encrypt and will make it accountable and traceable. Application of role-based access control (RBAC) is carried out through Azure Active Directory (AAD) to make sure that only users and services that are authenticated are able to communicate with the storage and the database. Also, data masking and transparent data encryption (TDE) is made available in Azure SQL to secure sensitive metadata being in rest. Both the database and file storage back up policies are established through Azure Recovery Services. The architecture is also hybrid-storage which is more durable, secure, and scalable without the need to use third party storage services such as S3. It is also adhering to industry best practice of securing file retention and audit capabilities. This layer alone rounds out the encrypted file lifecycle; what mobile devices have secure, encrypted storage, to ensure secure storage in the cloud with your customized, encrypted mobile cloud infrastructure; from encryption on the mobile device with the files stored in binary files at rest in the secure location, to encrypted meta data in the cloud, with your custom ChaCha20 + ECC hybrid model, this layer does it all.

6 Evaluation

6.1 Performance Metrics Used

To evaluate the efficiency of the proposed ChaCha20-ECC hybrid encryption framework, two key performance metrics were used: encryption time and decryption time as shown in Table 2, both measured in milliseconds. Encryption time refers to the total duration required to convert plaintext into ciphertext using the ChaCha20 cipher after establishing a secure session key via ECC. Decryption time measures the time taken to accurately retrieve the original file from the ciphertext using the same symmetric key. These metrics were chosen to reflect the real-world responsiveness and suitability of the system for mobile cloud environments, where speed and efficiency are critical.

Table 2: Performance

Metric	**Description**
Encryption Time (ms)	Time taken to encrypt the file using ChaCha20 after ECC key derivation
Decryption Time (ms)	Time required to decrypt the file on the cloud side using the derived session key

6.2 Evaluation Results

Table 3 shows the encryption and decryption times for JPG image files of increasing sizes using the ChaCha20-ECC hybrid model.

Table 3: Encryption and Decryption Time for JPG Files

File Size	Encryption Time (ms)	Decryption Time (ms)
2 MB	1219	217
5 MB	1483	342
10 MB	1927	850

Figure 4 illustrates the encryption and decryption performance of the proposed ChaCha20-ECC hybrid encryption scheme for JPG files of sizes 2MB, 5MB, and 10MB. The chart shows that encryption time increases steadily with file size, from 1219 ms for 2MB to 1927 ms for 10MB. Decryption time follows a similar upward trend, peaking at 850 ms for 10MB. The best performance is observed for 2MB JPG files, where encryption and decryption are achieved in just 1219 ms and 217 ms respectively. This figure confirms that the system maintains consistent scalability with increasing file size.

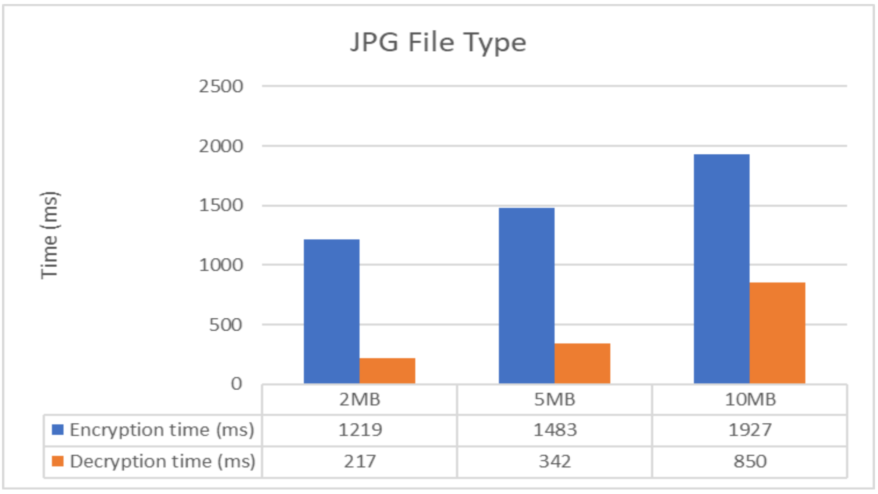

Figure 4: Encryption and Decryption Times for JPG File Type (Bar Chart)

Table 4 presents the performance metrics for PDF files, highlighting variations in processing time with file size.

Table 4: Encryption and Decryption Time for PDF Files

File Size	Encryption Time (ms)	Decryption Time (ms)
2 MB	1257	79
5 MB	1169	286
10 MB	1381	340

Figure 5 presents a bar chart comparing encryption and decryption times for PDF files of sizes 2MB, 5MB, and 10MB using the ChaCha20-ECC hybrid model. The encryption times are relatively consistent, ranging from 1169 ms to 1381 ms, while decryption times show a more significant rise from 79 ms at 2MB to 340 ms at 10MB. The best performance is observed for the 2MB file, achieving the lowest decryption time of just 79 ms. This indicates that the proposed encryption scheme is particularly efficient with smaller structured documents such as PDFs. Overall, performance scales well with file size.

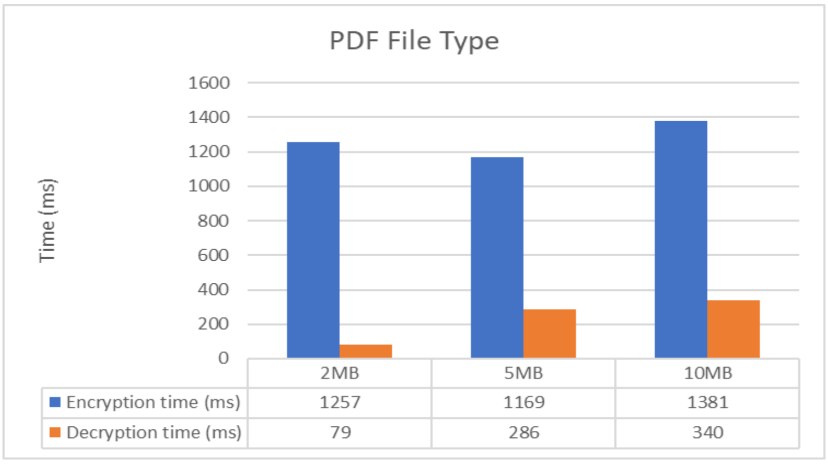

Figure 5: Encryption and Decryption Times for PDF File Type (Bar Chart)

Table 5 displays encryption and decryption times for DOC files, demonstrating consistent performance across sizes.

Table 5: Encryption and Decryption Time for DOC Files

File Size	Encryption Time (ms)	Decryption Time (ms)
2 MB	1267	117
5 MB	1223	162
10 MB	1596	394

Figure 6 shows the encryption and decryption times for DOC files of 2MB, 5MB, and 10MB using the proposed ChaCha20-ECC hybrid model. Encryption times range from 1223 ms to 1596 ms, while decryption times increase gradually from 117 ms to 394 ms as file size grows. The best performance is observed with the 2MB DOC file, which achieves fast processing—1267 ms for encryption and just 117 ms for decryption. The chart highlights consistent and scalable performance across all sizes, making the system suitable for document-heavy cloud applications where quick, secure transmission is essential.

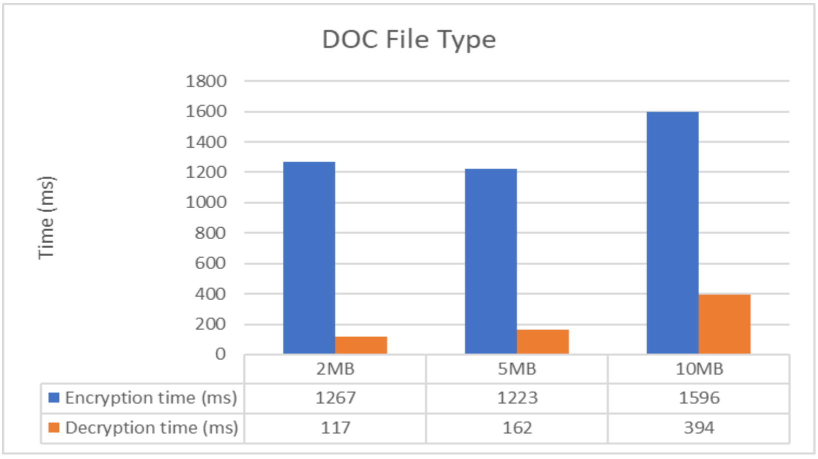

Figure 6: Encryption and Decryption Times for DOC File Type (Bar Chart)

Table 6 presents the encryption throughput (in MB/s) for JPG, PDF, and DOC files of various sizes using the ChaCha20-ECC model. It shows that throughput increases with file size, peaking at 7.24 MB/s for PDF (10MB), confirming efficient and scalable performance.

Table 6: Throughput Table

File Type	**File Size (MB)**	**Encryption Time (ms)**	**Throughput (MB/s)**
JPG	2	1219	1.64
JPG	5	1483	3.37
JPG	10	1927	5.19
PDF	2	1257	1.59

PDF	5	1169	4.28
PDF	10	1381	7.24
DOC	2	1267	1.58
DOC	5	1223	4.09
DOC	10	1596	6.26

6.3 Analysis of Results

The performance evaluation of the proposed ChaCha20-ECC hybrid encryption framework was conducted across three file types—JPG, PDF, and DOC—each tested at file sizes of 2MB, 5MB, and 10MB. The results show a consistent increase in encryption and decryption time as file size increases, which aligns with expectations for a stream cipher like ChaCha20. Among all tested files, PDF (2MB) achieved the lowest decryption time of just 79 ms, while JPG (10MB) showed the highest encryption time at 1927 ms.

Throughput analysis further strengthens the efficiency claims of the proposed model. The throughput peaked at 7.24 MB/s for PDF (10MB), indicating that the system scales well for larger files. Even with a 2MB file, throughput remained above 1.5 MB/s, proving the model's suitability for real-time mobile encryption. Across all file types, throughput increased with file size due to improved cipher stream utilization and reduced key setup overhead per byte.

Overall, the ChaCha20-ECC hybrid model demonstrates consistent, scalable, and high-throughput performance. The balance of ECC's secure key exchange with ChaCha20's fast encryption makes this model ideal for mobile cloud applications where low latency and strong encryption are both essential.

7 Conclusion and Future Work

7.1 Conclusion

This research introduces a novel hybrid cryptographic framework combining Elliptic Curve Cryptography (ECC) for secure key exchange and ChaCha20 for fast, lightweight data encryption, designed specifically for mobile cloud environments. The model addresses critical challenges such as latency, key management, and data confidentiality, while maintaining low computational overhead—making it ideal for mobile and IoT contexts. Experimental results using real-world file types—JPG, PDF, and DOC—at varying file sizes demonstrate the model's scalability and efficiency. Encryption times increased linearly with file size, while decryption remained significantly faster, affirming ChaCha20's stream cipher efficiency.

Additionally, throughput analysis confirmed the system's processing capabilities, achieving a peak throughput of 7.24 MB/s for 10MB PDF files and maintaining over 1.5 MB/s even for smaller 2MB files. These values validate the model's real-time performance and low-latency behavior. Integration with Microsoft Azure Cloud, ASP.NET Core backend, and Azure SQL Database ensured secure, scalable storage and communication. The solution adheres to modern cryptographic standards and preserves end-to-end encryption, ensuring data remains protected throughout its lifecycle.

In conclusion, the ChaCha20-ECC hybrid model offers a robust balance of security, speed, and scalability, making it highly suitable for secure mobile cloud applications in domains like healthcare, finance, and enterprise mobility.

7.2 Limitations

The proposed ChaCha20-ECC hybrid encryption model presents encouraging outcomes, there are various limitations that might touch on its implementation in some respects. To begin with, there is no user level authentication or role based access control in the mobile application layer of the system, though this might be required in multi user systems. Secondly, the encryption and the key generation pipeline, although effective, can only occur serially, inhibiting parallelism which can further speed up the time it takes in an instance of high-throughput. Also, ECC key management, although being lightweight compared to RSA, adds complexity when managing sessions, which can be even aggravated when information about key rotation or revocation capabilities should be present. Another assumption made by the prototype is that there are good network connections to facilitate secure communication between the mobile client and cloud back-end, which is not always stable in reality. In addition, integration of file integrity checking beyond simpler file hash is not complete e.g. digital signatures or zero-knowledge proofs were explored but de-scoped as outside the scope of file integrity checking. The second problem is the absence of UI/UX optimization on the mobile side because the existing interface deals only with the bare essentials. Finally, massive stress testing using multiple users simultaneously or in real time streams of files have not been done yet. Those limitations point at the possible areas of future improvement to bring the system to the production level.

7.3 Future Works

The existing framework of such a hybrid ChaCha20 + ECC encryption scheme, a number of future extensions are foreseen. Such a promising avenue is the inclusion of next-generation identity solutions and multi-factor authentication (MFA) to increase user-level protection pertaining to mobile applications. The use of post-quantum cryptography algorithms, and the ECC together could enhance the shield against future quantum cyberattacks. The next possible enhancement is the possibility to support parallel operations in both batch encryption and decryption to minimize latency in the high-traffic cases. Subsequent versions can include blockchain-based integration, which provides a record of key exchanges as well as audit trails that cannot be tampered with. Regarding the cloud side, the current implementation can be improved regarding compliance and scalability by extending it to contain policies of encryption on the Azure Blob Storage and data loss prevention (DLP) systems. Incorporation of automated key rotation and revocation mechanisms will solve the existing constraints in respect to cryptographic security to the session level. The mobile application can be encapsulated with the live file preview, usage statistics, and UI/UX optimization as well, to provide a better experience to the user. Media security applications could be supported through adding support of more file format (e.g., video, audio) and embed watermarking or fingerprinting capabilities. Lastly, in a future development, a comparison with other hybrid models such as RSA-AES or ECC-AES can be made to prove to be better faster. All of these improvements would have rendered the system much stronger, scaled, and enterprise-grade deployable.

References

Adeniyi, A.E., Abiodun, K.M., Awotunde, J.B., Olagunju, M., Ojo, O.S., Edet, N.P., 2023. Implementation of a block cipher algorithm for medical information security on cloud environment: using modified advanced encryption standard approach. Multimed. Tools Appl. 82, 20537–20551. https://doi.org/10.1007/s11042-023-14338-9

Ajagbe, S.A., Adeniji, O.D., Olayiwola, A.A., Abiona, S.F., 2024. Advanced Encryption Standard (AES)-Based Text Encryption for Near Field Communication (NFC) Using Huffman Compression. SN Comput. Sci. 5, 156. https://doi.org/10.1007/s42979-023-02486-6

Alkadi, O., Moustafa, N., Turnbull, B., 2020. A Review of Intrusion Detection and Blockchain Applications in the Cloud: Approaches, Challenges and Solutions. IEEE Access 8, 104893–104917. https://doi.org/10.1109/ACCESS.2020.2999715

Awan, I.A., Shiraz, M., Hashmi, M.U., Shaheen, Q., Akhtar, R., Ditta, A., 2020. Secure Framework Enhancing AES Algorithm in Cloud Computing. Secur. Commun. Netw. 2020, 1–16. https://doi.org/10.1155/2020/8863345

Hazra, R., Chatterjee, P., Singh, Y., Podder, G., Das, T., 2024. Data Encryption and Secure Communication Protocols:, in: Goel, P.K. (Ed.), Advances in Web Technologies and Engineering. IGI Global, pp. 546–570. https://doi.org/10.4018/979-8-3693-6557-1.ch022

Huang, Y., Xu, H., Gao, H., Ma, X., Hussain, W., 2021. SSUR: An Approach to Optimizing Virtual Machine Allocation Strategy Based on User Requirements for Cloud Data Center. IEEE Trans. Green Commun. Netw. 5, 670–681. https://doi.org/10.1109/TGCN.2021.3067374

Kai, C., Zhou, H., Yi, Y., Huang, W., 2021. Collaborative Cloud-Edge-End Task Offloading in Mobile-Edge Computing Networks With Limited Communication Capability. IEEE Trans. Cogn. Commun. Netw. 7, 624–634. https://doi.org/10.1109/TCCN.2020.3018159

Kurunthachalam, A., Ahmed, N., 2025. SECURE USER AUTHENTICATION AND DATA SHARING FOR MOBILE CLOUD COMPUTING USING MD5 AND ELGAMAL ENCRYPTION.

Maria Navin, J.R., M Lutimath, N., n.d. AN ENHANCED AES-ECC MODEL FOR THE SECURITY OF MOBILE APPLICATIONS USING CLOUD COMPUTING.

Muheidat, F., Tawalbeh, L., 2021. Mobile and Cloud Computing Security, in: Maleh, Y., Shojafar, M., Alazab, M., Baddi, Y. (Eds.), Machine Intelligence and Big Data Analytics for Cybersecurity Applications, Studies in Computational Intelligence. Springer International Publishing, Cham, pp. 461–483. https://doi.org/10.1007/978-3-030-57024-8_21

Pallavi, G.B., Jayarekha, P., 2022. Secure and efficient multi-tenant database management system for cloud computing environment. Int. J. Inf. Technol. 14, 703–711. https://doi.org/10.1007/s41870-019-00416-5

Prakash, V., Goyanka, T., Sharma, S., Garg, L., Shukla, V., 2024. Secure Text Transfer Using Diffie–Hellman Key Exchange Algorithm in Cloud Environment, in: Chaturvedi, A., Hasan, S.U., Roy, B.K., Tsaban, B. (Eds.), Cryptology and Network Security with Machine Learning, Lecture Notes in Networks and Systems. Springer Nature Singapore, Singapore, pp. 631–643. https://doi.org/10.1007/978-981-97-0641-9_43

Shabbir, M., Shabbir, A., Iwendi, C., Javed, A.R., Rizwan, M., Herencsar, N., Lin, J.C.-W., 2021. Enhancing Security of Health Information Using Modular Encryption Standard in Mobile Cloud Computing. IEEE Access 9, 8820–8834. https://doi.org/10.1109/ACCESS.2021.3049564