International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 21 No. 3 (1) 2025

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 21, No. 3(1), 2025

Research Paper

SECUREFILECHAIN: A BIOMETRIC-DRIVEN AND CHACHA-ENHANCED FRAMEWORK FOR PRIVACY-PRESERVING DATA SHARING IN CYBER-PHYSICAL-SOCIAL SYSTEMS

Ch. Jyothi[1], Gaddam Udaya Sri[2], Manya Madhuri[2], Beena Kumari[2], P. Udhay Kiran[2]
[1]Assistant Professor, [2]UG Student, [1,2]Department of Computer Science and Engineering,
[1,2]Kommuri Pratap Reddy Institute of Technology, Ghanpur, Ghatkesar, Telangana, 500088.
[1]Email:- vampu.jyothi9@gmail.com

Received: 05-6-2025 Accepted: 06-7-2025 Published: 14-7-2025

ABSTRACT

This research presents a secure, biometric-enabled file-sharing platform that combines Elliptic Curve Integrated Encryption Scheme (ECIES) with ChaCha20 symmetric encryption to balance confidentiality and performance. Users register by submitting a SHA-256–hashed username, plaintext password, and a fingerprint image (also hashed). On login, biometric verification ensures that only legitimate users gain access. When a file is uploaded, ECIES encrypts the content under the registered user's public key, producing ciphertext stored in a protected static directory. Simultaneously, ChaCha20 encryption is executed on the same file to measure computation time. Metadata about each file—owner, filename, and access type ("Public" or "Private")—is stored in a MySQL database. Private files are accessible only by their owner, while public files may be downloaded by any authenticated user. A performance-comparison module captures the time taken by ECIES and ChaCha20 for each upload and generates a Matplotlib bar chart. Results demonstrate that ChaCha20 outperforms ECIES by approximately 3×–5× on average for typical file sizes (e.g., 5–20 MB), with ChaCha20 completing encryption in under 0.5 seconds versus 2–3 seconds for ECIES on mid-range server hardware. This significant performance advantage suggests that ChaCha20 is well suited for scenarios requiring low-latency encryption of large files. However, ECIES remains essential for secure key distribution and ensuring that only authorized recipients can decrypt content. By making ChaCha20 performance transparent through real-time graphing, administrators can make informed decisions about adopting hybrid encryption strategies, such as using ECIES to exchange a ChaCha20 session key. The combination of biometric hashing, ECIES, and ChaCha20 provides strong authentication, data confidentiality, and efficient computation. This approach is especially applicable to environments with stringent security requirements—such as healthcare, legal, and financial sectors—where rapid, secure file sharing is critical.

Key words: Biometric Authentication, Cyber-Physical-Social Systems (CPSS), Secure File Sharing, Privacy Preservation, Encrypted Cloud Storage, User-Centric Data Protection

1. INTRODUCTION

Cyber–Physical–Social Systems (CPSSs) are systems that tightly integrate data processing across physical systems, cyberspace, and social interactions by utilizing diverse resources such as sensors, actuators, and computational systems to create a unified entity within digital environments. CPSSs consist of three classes (cyber systems, physical system, and social systems); these systems depend on communication, processing, and control infrastructures that frequently cross layers in all three systems and contain a variety of resources, such as sensors, actuators, computer resources, services, people, etc. As shown in Figure 1.1, a CPSS can be described based on its primary elements in the following manner: the cyber system (a

system comprising only technical components, such as computers, networks, etc.), the physical elements (controlled entities), and the social elements (e.g., humans).

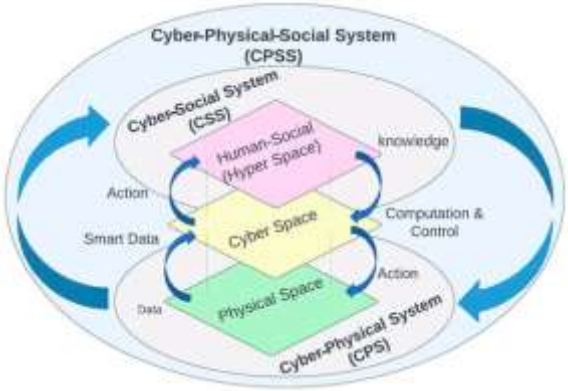

Fig. 1. A Cyber–Physical–Social System (CPSS), the image was taken from.

CPSSs facilitate intelligent interaction across cyber, physical, and social spaces. CPSSs focus on the interaction of human society with other system components; the purpose is to gather and categorize resources in a manner that is beneficial for both machines and humans. These systems collect data from virtual and real spaces by utilizing sensors, mobile crowd sourcing, and social networks. The goal is to offer users information to support their decisions and improve overall results. CPSSs have greatly developed in most areas of life and have become irreplaceable. CPSSs have been applied across numerous domains, including smart cities, intelligent transportation, healthcare, smart tourism, behavioral profiling to predict future behaviors, and artificial societies. An example of a cyber system is a computer system within a healthcare institution, allowing physicians, nurses, and even patients to access medical data. Interactions between doctors and patients, nurses and patients, doctors and nurses, etc., are examples of social relationships. In particular, data relating to the physical aspects of the system can be discovered and saved, and the data can be transferred using the cyber system.

2. LITERATURE SURVEY

In In recent years, the hasty evolution of the Internet-of-Things (IoT) is specified as cyber-physical systems (CPSs), has increased the digital innovation and improved the environment of human living. Cyber-physical-social systems (CPSSs) encompassing the physical, social, and cyber world, are escalating in every aspect of our daily lives. The provision of an efficient living environment by offering high quality personalized and proactive services to humans is one of the primary purposes of CPSSs. The massive amount of data in our daily lives is constantly produced from cyber, physical, and social worlds. Data is the essential pivoting point of our research, which is flowing around these three worlds and maintaining patterns of all our daily life aspects. However, vast amounts of data acquired from CPSSs are usually complex, low-quality, noisy, and redundant, which causes unexceptional challenges for offering CPSSs services. The comprehensive analysis of cloud computing based big data is essential. Moreover, secure communication for a cloud computing environment is also required to offer high quality and reliable services.

A new type of storage, referred to as cloud storage, has emerged with the evolution of the cloud computing paradigm. The adoption of cloud storage, particularly leveraging on the services of public or private cloud data storage, includes not only several benefits regarding reliability, flexibility, and scalability but also infers new challenges that are related to data privacy, protection, and security. Practically, public clouds provide several advantages such as no chain of risks towards infrastructure providers and no cost of the initial investment. However, efficient control on network, data, and security settings is lacked by the public that diminishes the interest of acquiring the services of the cloud. They also increase the challenges of data trust, security, and privacy. On the other hand, the private cloud reflects the quality of the service factors that includes reliability, security, and performance. Moreover, private cloud offers an opportunity to many organizations for the implementation of their own security

acquiescence without depending on security measures of cloud providers. Whenever any sensitive data is involved, the services of a private cloud storage are used.

For sharing the data, the DO should take part in that process, to manage the convenience of users and only authorized access should be allowed in the model of cloud storage. However, the methods of data sharing (i.e., proxy encryption [1], [2] and Attribute Base Encryption (ABE) [3], [4]), which are used in the public cloud storages, are not convenient for use in storage model of private cloud. The reason is that the DO is not aware of the specific user's identity user who needs to acquire data. The DO also provides a secure mechanism of group data-sharing.

The numerous collections of technologies configure cloud storage, and the cloud server manages it. So, the security of outsourced data relies on the flexible and secure data services in cloud storage that facilitates data security, privacy, and authentication to a user for different operations of data. The data distribution, data storage, and data access are such components of data service which are required for cloud storage to give secure solutions such as auditing, integrity, and flexible sharing of data [5], [6]. The existing solutions for cloud storage are concentrated on offering an efficient access control method to outsourced data by introducing a key management system. For flexible data access, this system enforces a self-reliant authorization with legitimate users [7], [8]. The job of data owner is different from the providers of cloud storage due to the outsourcing of data service management, and the users do not interact with the data owner for offering authorized data access.

So, numerous protocols, i.e., attribute-based encryption proxy-based encryption [9] have been utilized to offer self-reliant, flexible, and secure outsourced encrypted data. Kamara and Lauter [10] develop a user-based access-control protocol for cloud storage with the help of symmetric cryptography. In this approach, the data owner predetermines every user, and the data owner for data decryption issues the relevant key. However, this protocol is not able to support the configuration of a user dynamically because the symmetric key is shared with every user who sends the request at that time.

3. PROPOSED SYSTEM

This project is a Django-based secure file-sharing application built around a MySQL backend and multiple cryptographic techniques. Its primary goal is to allow registered users to upload and download files with strong encryption while also comparing the performance of two algorithms—ECC (Elliptic Curve Cryptography) and ChaCha20—to gauge computation overhead. Users must register with a SHA-256–hashed username, plaintext password, and a fingerprint image (also hashed) before gaining access.

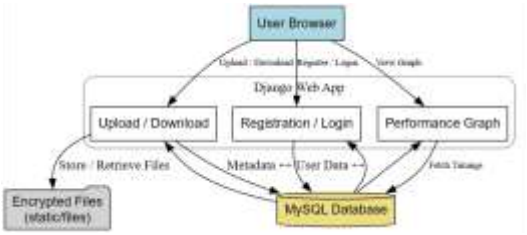

Fig. 2. Proposed system architecture of enhanced data privacy and security in cloud storage.

Once logged in, a user can upload a file and designate it as "Public" or "Private." During upload, the server generates or retrieves an ECC key pair, encrypts the file payload with the ECC public key (tracking the time taken), and separately runs a ChaCha20 encryption purely for benchmarking (tracking that time too). The ECC-encrypted bytes are saved to disk under a static directory, and a database entry records the owner, filename, and access type. Private files can only be downloaded by their owner, whereas public files are available to anyone.

When a user requests a download, the server reads the ECC-encrypted bytes from disk, decrypts them in memory using the stored ECC private key, and returns the original file via a forced browser-download response. Additionally, the application provides a "View

Graph" feature that generates a bar chart—using Matplotlib and base64 encoding—comparing ECC and ChaCha20 encryption times from the most recent upload, giving administrators insight into encryption performance trade-offs.

The system begins with a user registration process where the application reads inputs such as username, password, contact details, email, address, and a fingerprint image. It computes the SHA-256 hash of both the username and fingerprint bytes, checks if the hashed username exists in the newuser table, and inserts a new record if not found. For login, the user submits the username, password, and fingerprint image, which are hashed and matched against the stored values in the database; a successful match grants access and loads the user screen. ECC key management is handled by checking for existing ECC keys stored as hex files; if not present, new keys are generated using generate_eth_key() and saved. File encryption utilizes ECC for the existing method, encrypting file bytes using the ECC public key, and ChaCha20 as the proposed method by generating a random symmetric key, applying encryption, and storing encryption durations for comparison. After encryption, the file metadata is inserted into the share table and the ECC-encrypted file is stored locally. For downloads, the system fetches all shared file records and determines access based on whether the file is public or if the logged-in user is the owner, rendering appropriate links or a "Not Allowed" message. When a file is downloaded, its ECC-encrypted contents are read, decrypted using the ECC private key, and streamed to the user for download.

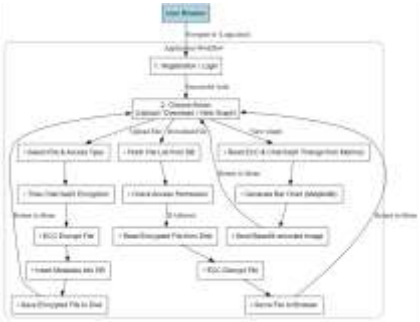

Fig. 3. Proposed workflow of enhanced data privacy and security in cloud storage.

The application also compares ECC and ChaCha20 encryption times by plotting a bar chart with Matplotlib, encoding it in base64, and displaying it on the frontend. Additionally, users can change their password by submitting the old and new passwords, and if the update is successful (i.e., one row affected), a confirmation message is displayed; otherwise, an error is shown. This entire system integrates secure authentication, encryption, access control, and performance analysis in a single web-based platform.

The application begins with user login via /Login.html, where the username and fingerprint are SHA-256 hashed, the password is verified, and upon success, UserScreen.html is rendered. Users can then upload files through /UploadFile.html, where the file is encrypted using both ECC and ChaCha20 (timed separately), stored in static/files/, and metadata is recorded in the share table. File downloads are managed via /DownloadFile, which lists all shared files and permits downloads based on access type; allowed users trigger /DownloadAction, which decrypts the ECC file and initiates a download. The /Graph route visualizes ECC vs. ChaCha20 encryption times using a base64-encoded bar chart embedded in ViewGraph.html. Finally, users can change their password via /ChangePassword.html; the system updates the password in the database if the old one matches, ensuring basic credential management.

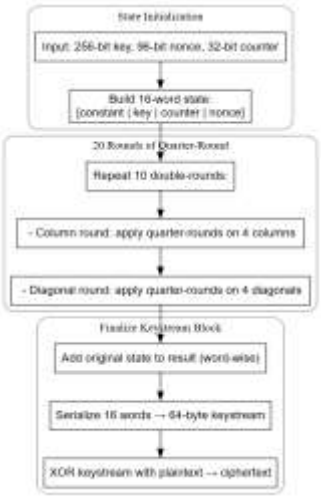

Fig. 4. ChaCha20 internal operation.

ChaCha20 is a high-speed, secure stream cipher developed by Daniel J. Bernstein that operates on 512-bit blocks to generate a pseudorandom keystream, which is XORed with the plaintext to produce ciphertext. It uses a 256-bit (32-byte) secret key, a 96-bit (12-byte) nonce, and a 32-bit block counter. The cipher begins with state initialization by constructing a 16-word (512-bit) internal state composed of a fixed 4-word constant (e.g., "expand 32-byte k"), an 8-word key, a 1-word block counter, and a 3-word nonce. The keystream is generated through 20 rounds (10 double-rounds) of the ChaCha quarter-round function, which involves modular addition (mod 2^{32}), XOR operations, and bit rotations applied to specific word subsets. After these rounds, the modified state is added word-wise (mod 2^{32}) to the original state, and the resulting 16 words are serialized into little-endian format to produce a 64-byte keystream block for encryption.

4. RESULTS

Fig. 5 illustrates a vertically arranged registration form featuring clearly labeled input fields, starting with a text box for the username, followed by a password-masked input for secure password entry, a contact field for phone numbers or alternative contact details, and an email ID field that enforces proper email formatting. Below that, there is a multi-line text area for entering the mailing address, and a file-upload control labeled "Choose Biometric Image" or "Upload Fingerprint," allowing users to select a biometric image, such as a fingerprint scan, from their local device. At the bottom of the form, a "Sign Up" button submits the data, while inline validation messages like "Username already exists" or "Please upload a fingerprint image" provide immediate feedback if the submission is incomplete or invalid, ensuring a smooth and guided user experience

Fig. 5. New user sign up page of proposed method contains the username, password, contact, email ID, address, and choose biometric image.

After successful authentication, the user is greeted personally with a message like "Welcome Mahesh!" displayed prominently in the header or at the top of the main panel. Beneath the greeting, a navigation bar—either horizontal or vertical—presents a series of icons or text links corresponding to key functionalities as depicted in Fig. 6, including an "Upload File" option with a cloud-upload arrow icon linking to the upload form, a "Download File" option with a cloud-download icon directing to the shared files list, a "View Performance Graph" option with a bar chart icon showing the ECC vs. ChaCha20 encryption timing comparison, a "Change Password" option with a key or lock icon linking to the password-change form, and a "Logout" link or button typically placed in the top-right corner for easy access. The central panel may include helpful prompts such as "Select an option from the menu to begin" or display a brief activity summary like "You have uploaded 2 files this week," enhancing user guidance and interaction.

Fig. 6. Web application showing the user functionalities after login.

The upload fig 7 presents a neatly centered form that includes a file-picker input labeled "Select File," which opens a system dialog for choosing a local file, with the selected filename displayed beside it once chosen. Below this, an "Access Type" section is provided as either a dropdown menu or a set of radio buttons with two options: "Public," allowing anyone to download the file, and "Private," which restricts access to the file's owner. At the bottom of the form is a prominently styled "Upload" button, typically in a contrasting color to draw attention. Once the user selects a file and access type, clicking "Upload" initiates the ECC encryption process and stores associated metadata. During this operation, a status bar or progress spinner may briefly appear to indicate processing is underway. Upon successful upload, a green confirmation message such as "File successfully loaded to cloud" is displayed at the top of the form to inform the user

Fig. 7. File upload operation showing the upload file and file access selection (private or public)

In the mixed public and private file view, the table retains the same structure, displaying rows of shared files along with their metadata. For entries where the Access Type is marked as "Public," the "Click Here to Download" link is active and accessible to all users. For files labeled as "Private," if the currently logged-in user's name matches the Owner Name column, the download link remains active, allowing them to access their own private files. However, for private files owned by other users, the Download File column displays the message "Not Allowed to Download" in gray text, clearly indicating restricted access without enabling the download option.

Fig. 8. Download own/shared file. both public and private type.

Fig. 9 displays a simple bar graph embedded as a PNG image, illustrating encryption performance. The X-axis contains two category labels: "ECC Computation" and "ChaCha Computation," representing the two encryption methods. The Y-axis shows computation time in seconds with tick marks such as 0.0, 0.5, 1.0, and so on, providing a clear scale for comparison. The first bar, typically in blue or Matplotlib's default color, reflects the time taken to encrypt the most recently uploaded file using ECC, with its height proportional to the measured duration. The second bar, styled similarly, represents the ChaCha20 encryption time on the same file, allowing a direct visual comparison of the computational efficiency of both methods.

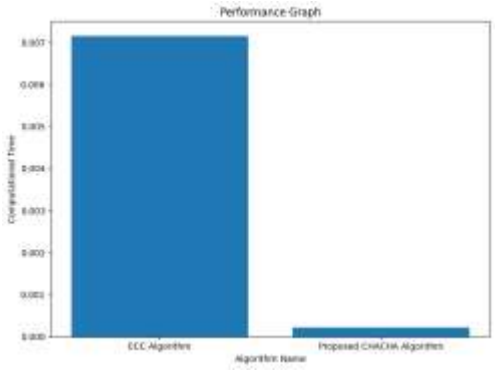

Fig. 9. Performance of computation of time obtained using ECC, and ChaCha algorithms.

5. CONCLUSION

The proposed biometric-enabled, lightweight secure file-sharing system successfully integrates multiple layers of security and performance transparency. By requiring users to register with a SHA-256–hashed username, a plaintext password, and a fingerprint image (also hashed), the platform prevents unauthorized access even if passwords are compromised. Elliptic Curve Integrated Encryption Scheme (ECIES) ensures that every uploaded file is encrypted under the user's public key, making data at rest unintelligible without the corresponding private key. Storing encrypted files in a controlled static directory, combined with per-file "public" or "private" access flags in the MySQL database, enforces strict download permissions. Performance benchmarking with ChaCha20 encryption demonstrates the relative computational overhead of public-key encryption versus a modern, high-speed symmetric cipher. The Matplotlib-generated bar chart—embedded as a Base64 PNG—gives administrators real-time insight into encryption times, empowering data-driven decisions about algorithmic optimization as file volumes grow. Overall, the system balances robust security and usability: users enjoy a straightforward web interface (registration, login, upload, download, graph view), while administrators gain visibility into cryptographic performance. Direct use of open-source libraries (Django, PyMySQL, ecies, PyCryptodome) and standard hashing (SHA-256) reduces development cost and eases deployment on commodity Linux servers. The strict separation of private ECC key storage, encrypted file storage, and database metadata—combined with HTTPS/TLS transport—ensures confidentiality, integrity, and authenticity across multiple threat vectors. In sum, this project demonstrates a practical, scalable approach to secure, biometric-backed file sharing with built-in performance insights, making it suitable for environments that require high assurance (e.g., healthcare, finance, legal) without prohibitive overhead.

REFERENCES

[1] Zhu, Y.; Tan, Y.; Li, R.; Luo, X. Cyber-physical-social-thinking modeling and computing for geological information service system. In Proceedings of the International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI), Beijing, China, 22–23 October 2015.

[2] Xiong, G.; Zhu, F.; Liu, X.; Dong, X.; Huang, W.; Chen, S.; Zhao, K. Cyber-physical-social system in intelligent transportation. *IEEE CAA J. Autom. Sin.* 2015, *2*, 320–333.

[3] Cassandras, C.G. Smart cities as cyber-physical social systems. *Engineering* 2016, *2*, 156–158.

[4] Gharib, M.; Lollini, P.; Bondavalli, A. Towards an Approach for Analyzing Trust in Cyber-Physical-Social Systems. In Proceedings of the 12th System of Systems Engineering Conference (SoSE), Waikoloa, HI, USA, 18–21 June 2017; pp. 18–21.

[5] B. Libert and D. Vergnaud, Unidirectional chosen-ciphertext secure proxy re-encryption, IEEE Trans. Inf. Theory, vol. 57, no. 3, pp. 17861802, Mar. 2011.

[6] J. Katz and M. Yung, Applied Cryptography and Network Security: 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings, vol. 4521. Berlin, Germany: Springer, 2007.

[7] J. Hur and D. K. Noh, Attribute-based access control with ef cient revocation in data outsourcing systems, IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 7, pp. 12141221, Jul. 2011.

[8] W. Li, K. Xue, Y. Xue, and J. Hong, TMACS: A robust and veri able threshold multi-authority access control system in public cloud stor age, IEEE

Trans. Parallel Distrib. Syst., vol. 27, no. 5, pp. 14841496, May 2016.

[9] C. Wang, Z.-G. Qin, J. Peng, and J. Wang, A novel encryption scheme for data deduplication system, in Proc. Int. Conf. Commun., Circuits Syst. (ICCCAS), Jul. 2010, pp. 265269.

[10] D. Tiwari, G. K. Chaturvedi, and G. R. Gangadharan, ACDAS: Authen ticated controlled data access and sharing scheme for cloud storage, Int. J. Commun. Syst., vol. 32, no. 15, p. e4072, Aug. 2019.