AMERICAN JOURNAL OF MANAGEMENT AND IOT MEDICAL COMPUTING
Peer Reviewed, Referred & Indexed Journal
E-ISSN: 3069-0110
Vol.5, No.2(1) (2026)
www.ajmimc.com

Privacy-Preserving Cryptographic Channel Design for Distributed Cross-Domain Data Transfer Networks

K. Anusha Reddy[1*], K. Vamshee Krishna[1*], Morla Manish[2], Putla Anupam Kumar[2], Payam Aishwarya[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, 501301, Telangana, India.

*Correspondence: K. Anusha Reddy (anusha.kundanapally@gmail.com), K. Vamshee Krishna (vamshik825@gmail.com)

ABSTRACT

The increasing adoption of distributed systems has created a strong demand for secure and efficient data exchange, exposing the limitations of traditional centralized architectures. Conventional file-sharing systems rely on a single server for authentication and storage, which leads to issues such as single-point failure, identity exposure, and limited scalability. These systems often lack robust identity verification across distributed nodes, making them vulnerable to impersonation, interception, and unauthorized access. Additionally, their dependence on static credential mechanisms and inefficient communication models results in higher latency and reduced system performance. Such limitations, including centralized identity management, increased processing overhead, and restricted collaboration between nodes, highlight the need for a decentralized and lightweight secure framework. To overcome these challenges, a system is developed using a Django-based web platform integrated with a Peer-to-Peer (P2P) architecture and Elliptic Curve Cryptography (ECC). The proposed system utilizes Elliptic Curve Diffie–Hellman (ECDH) as a key agreement mechanism to securely establish a shared secret between peers. When a requested file is not available locally, the system initiates secure ECC-based authentication with another peer and retrieves the file without relying on a central authority, ensuring confidentiality, integrity, and anonymity during communication. The Django framework effectively manages user interactions, database validation, file processing, and performance visualization, while the use of ECDH enables strong security with smaller key sizes, resulting in faster execution and improved overall efficiency.

Keywords: Decentralized Architecture, Elliptic Curve Diffie–Hellman (ECDH), Data Confidentiality, Django Framework, Secure Data Exchange.

1. INTRODUCTION

The continuous advancement of IoT has significantly transformed modern communication by enabling seamless interaction among devices, systems, and users across diverse environments without limitations of time or location. IoT applications are generally categorized into event-driven and time-driven models, where event-driven systems activate sensors only upon detecting specific conditions such as motion or object presence within a defined region, ensuring efficient and selective data transmission, whereas time-driven systems periodically collect environmental parameters such as temperature, humidity, and pressure and transmit them to centralized servers for continuous monitoring and analysis, as illustrated in Fig. 1.1. This periodic data acquisition plays a crucial role in applications requiring real-time insights and consistent data updates. However, the data generated by IoT devices often include sensitive and privacy-related information, making security a critical concern that must be addressed through robust mechanisms [1].

With the exponential growth in connected devices and the emergence of advanced communication technologies such as 6G, which offers significantly higher data rates and lower latency compared to 5G [2], IoT ecosystems are expected to scale to billions of interconnected devices, forming the backbone of applications such as telemedicine, smart homes, industrial automation, and intelligent transportation systems [3].

Despite these advancements, IoT devices are inherently constrained by limited processing power, memory, and energy resources, making it difficult to handle large volumes of heterogeneous data generated continuously. To overcome these limitations, data is often offloaded to cloud computing platforms, where powerful computational resources and scalable storage capabilities are available for efficient processing, analysis, and long-term storage. Cloud computing provides virtually unlimited resources [4], effectively compensating for IoT constraints while leveraging real-time data streams for advanced analytics and decision-making. However, the integration of IoT with cloud infrastructure introduces several security challenges, including data leakage, unauthorized access, and potential cyberattacks targeting centralized storage systems. These risks highlight the importance of implementing strong security measures such as encryption, secure communication protocols, and reliable authentication mechanisms, where user authentication plays a vital role in ensuring that only legitimate users can access system resources while maintaining data integrity, confidentiality, and overall system trustworthiness [5].

2. LITERATURE SURVEY

Zhong, et al. [6] introduced a secure data migration framework for cloud environments aimed at enabling trusted communication between different cloud service providers. Their approach is based on a mutual authentication and key agreement scheme that utilizes ECC-based certificate-free cryptography, eliminating the dependency on traditional certificate management systems. This significantly reduces complexity while maintaining strong security guarantees. The framework ensures secure cross-cloud data transfer by establishing trust between entities before data exchange. Performance evaluations demonstrated that the proposed scheme reduces both computational cost and communication overhead compared to conventional methods, making it suitable for large-scale and dynamic cloud environments where efficiency and security are critical. Tyagi, et al. [7] critically analyzed an existing authentication and key agreement protocol for wireless sensor networks and identified multiple security vulnerabilities, including susceptibility to offline password guessing attacks, weak session key protection, and vulnerability to man-in-the-middle attacks. To address these issues, they proposed an enhanced authentication scheme specifically tailored for IoT environments. The improved protocol strengthens user authentication, ensures secure session key establishment, and enhances resistance against various cyber threats. The robustness of the proposed scheme was validated using the RoR model, confirming its ability to provide secure communication while maintaining computational efficiency suitable for resource-constrained IoT devices. Chen, et al. [8] developed an anonymous authentication and key establishment protocol, known as FAuth, for smart grid environments. The protocol is designed using bilinear pairing and is based on the computational Diffie–Hellman problem to ensure secure communication between entities. A key contribution of their work is the modification of the registration process at the Key Generation Center, which prevents leakage of private keys and enhances overall system security. Additionally, the protocol is optimized to reduce computational overhead for smart meters and aggregators, making it practical for real-world deployment. The security of the scheme was formally verified using BAN logic and the Random Oracle Model, demonstrating its effectiveness against common security threats. Ali, et al. [9] examined the limitations of existing authentication mechanisms, including PKI-based, blockchain-based, and PUF-based approaches, highlighting issues related to high computational overhead and security vulnerabilities. They proposed a provably secure anonymous authentication protocol that achieves a balance between efficiency and strong security guarantees. The scheme minimizes communication, computation, and storage costs while ensuring resistance to various attacks such as insider attacks, eavesdropping, guessing attacks, and leakage of ephemeral secrets. The protocol was formally validated using BAN logic and semantic security analysis, demonstrating its robustness and suitability for secure information transmission in distributed environments.

Peivandizadeh, et al. [10] explored the expanding landscape of IoT applications, including smart cities, healthcare systems, and home automation, emphasizing the increasing need for secure communication across interconnected devices. Their study identified critical security requirements such as data confidentiality, node privacy, identity protection, and enforcement of security policies. The authors proposed a secure key exchange and authentication scheme designed to address these challenges while maintaining efficiency in resource-constrained environments. The study highlights that authentication plays a fundamental role in IoT security by ensuring that only legitimate devices can establish communication, thereby preventing unauthorized access and potential cyber threats.

Zhao, et al. [11] proposed a blockchain-based trust management and authentication framework for IoT environments, aiming to address issues related to unreliable data and lack of trust among interconnected devices. Their model stores trust values of participating entities within a blockchain, ensuring immutability and resistance to tampering. The framework also incorporates an anonymous authentication mechanism that enables secure identity verification without exposing sensitive user information. By supporting cross-domain interactions, the system enhances interoperability while maintaining privacy and security. The authors demonstrated that their approach improves decision-making accuracy in dynamic environments by ensuring that only trustworthy entities participate in communication. Wu, et al. [12] developed a lightweight authentication and key agreement protocol specifically designed for IoHT systems, where sensitive healthcare data is transmitted over potentially insecure networks. Their study identified several vulnerabilities in existing authentication mechanisms, particularly exposure to insider attacks and weaknesses in protecting patient data. The proposed protocol enhances security by incorporating efficient cryptographic operations that ensure data confidentiality and integrity while maintaining low computational overhead. This makes it suitable for resource-constrained medical devices such as wearable sensors and smart monitoring systems. The security of the protocol was validated through formal analysis, demonstrating its robustness against common attack scenarios. Deep, et al. [13] introduced a blockchain-based authentication protocol for cloud database systems, focusing on preventing unauthorized access and ensuring data integrity. Their framework leverages the immutable nature of blockchain to securely store authentication credentials, making it resistant to insider manipulation and data tampering. The system also provides traceability of user activities, enabling effective auditing and monitoring of access patterns. The authors validated their approach using the Scyther security analysis tool, which confirmed its resistance to various attacks, including replay attacks, impersonation attacks, and denial-of-service attacks. This work highlights the effectiveness of combining blockchain with traditional authentication mechanisms to strengthen cloud security.

Lee, et al. [14] developed a blockchain-based data access control and key agreement system tailored for IoT environments. Their approach ensures essential security properties such as non-repudiation, accountability, and secure key exchange between communicating entities. The system integrates smart contracts to automate access control decisions and enforce security policies without relying on centralized authorities. The authors conducted both formal and informal security analyses, along with performance evaluations, to demonstrate the feasibility and efficiency of their model. Comparative results showed that the proposed system achieves a balance between strong security guarantees and acceptable computational and communication overhead. Kairaldeen, et al. [15] investigated the scalability and performance optimization of blockchain-based communication systems in IoT networks. Their study focused on improving execution efficiency by integrating advanced data structures such as Merkle Mountain Hash Trees along with cryptographic techniques including SHA3 hashing and AES-128 encryption. The results demonstrated significant improvements in processing time and system scalability when handling large datasets and high transaction loads. Specifically, the proposed approach achieved notable performance gains compared to existing methods while maintaining strong data

integrity and security. The authors concluded that optimizing blockchain architectures is essential for enabling efficient and scalable IoT-based applications.

3. PROPOSED METHODOLOGY

The system is designed as a secure and scalable peer-to-peer file sharing framework that integrates a Django-based web application with a distributed P2P network and a backend database, enabling decentralized storage and secure data exchange, as illustrated in Fig. 1. The Django application acts as the central control layer, handling user interactions such as registration, login, file upload, and download, while validating credentials against the database to ensure secure authentication and controlled access. Once authenticated, users can upload files by selecting a target peer node, where the Django server reads the file content and transmits it using socket-based communication with serialized data transfer.

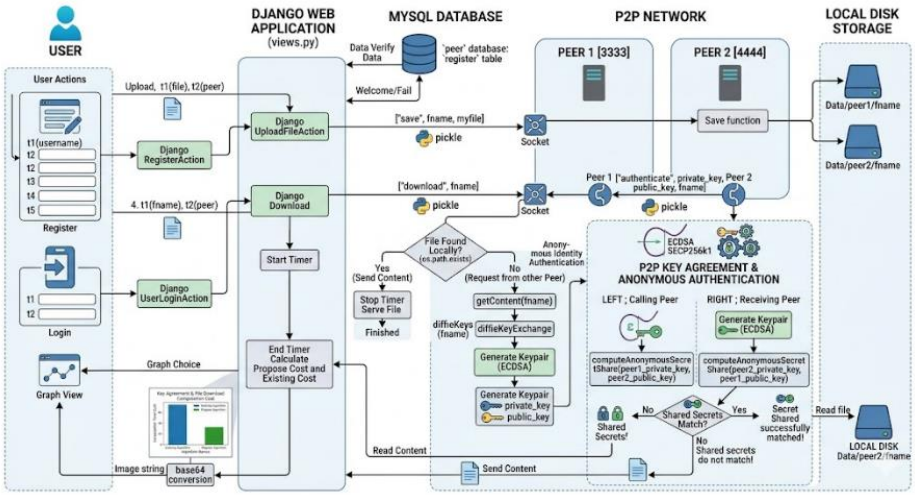

Fig. 1: Proposed system architecture.

The receiving peer stores the file in its local storage directory and sends an acknowledgment, ensuring decentralized file distribution and improved scalability by avoiding reliance on a central repository. During file download, the system first performs a local availability check at the selected peer node, allowing immediate retrieval if the file exists locally, thereby reducing latency and unnecessary network traffic. If the file is not available, the system initiates cross-peer communication, where the request is forwarded to other peers in the network to locate the file, retrieve it, and return it through the requesting peer to the user, ensuring high availability, redundancy, and fault tolerance. To secure inter-peer communication, the system incorporates an ECDH-based key agreement mechanism, where both communicating peers generate their own key pairs, exchange public keys, and independently compute a shared secret used to encrypt the file transfer process. This ensures secure and authenticated communication without exposing private keys, thereby maintaining confidentiality and anonymity. Additionally, the system continuously monitors performance by recording computation time during file transfer operations, allowing comparison between the proposed approach and existing methods. These performance metrics are visualized using graphical representations, enabling users and administrators to analyze system efficiency and improvements. By integrating secure authentication, decentralized storage, efficient peer communication, cryptographic key exchange, and performance evaluation, the system delivers a robust, transparent, and efficient solution for modern file sharing environments.

3.1 CRYPTOGRAPHY ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

Elliptic Curve Cryptography (ECC) is a public-key cryptography approach that provides strong security with much smaller key sizes than RSA or DH for the same level of protection. At its core, ECC uses arithmetic on points of an elliptic curve defined over a finite field; the hardness of the elliptic-curve discrete logarithm problem (ECDLP) i.e., given a point P and kP find k is what makes ECC secure.

Practically this means you can generate a private scalar (secret integer) and a corresponding public point (scalar × base point) and use those pairs to perform key agreement (ECDH) or digital signatures (ECDSA) with efficient computation and compact keys/packets ideal for constrained systems (IoT, mobile, or distributed peers). In the project ECC is used to perform an anonymous Diffie–Hellman-like key exchange between peers. Each peer generates an ephemeral keypair (private scalar and public point) when a secure exchange is needed. They share the public points (not private scalars) over the socket, then each peer multiplies its private scalar with the other peer's public point to compute a shared point; extracting a coordinate (e.g., the x-coordinate) or applying a KDF yields the shared secret. The peers compare secrets (or use them to derive symmetric keys) only if the computed secrets match is the requesting peer considered authenticated, and only then the file is served. This flow provides mutual key agreement without exposing private keys or fixed identity credentials.

3.2 PEER-TO-PEER NETWORKING (P2P)

Peer-to-Peer (P2P) networking is a decentralized communication architecture where each node called a *peer* acts as both a client and a server. Unlike traditional centralized systems where a single server manages all requests, P2P allows peers to directly communicate, store, and exchange data with each other. This design removes bottlenecks, reduces dependency on centralized servers, and improves system fault tolerance. In your project, Peer1 and Peer2 operate as independent nodes capable of storing and providing files on request without involving a central authority. In the proposed system, each peer runs as a multi-threaded socket server, continuously listening on its assigned port. The Django application communicates directly with these peers through socket connections to request file uploads, downloads, or authentication. Each peer contains its own storage directory (Data/peer1 or Data/peer2) where uploaded files are maintained. When a user requests a file, the selected peer checks if the file exists locally; if it does, the peer directly returns the file to Django. This design ensures efficiency and reduces retrieval time when files are distributed across nodes. If a peer does not have the requested file, the P2P network enables automatic peer-to-peer collaboration.

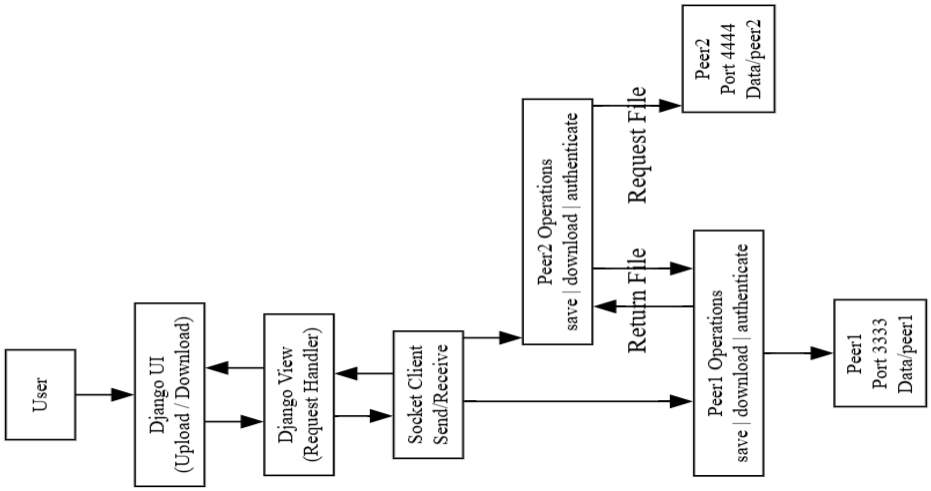

Fig. 2: Peer-to-Peer (P2P) Networking Architecture in the Proposed System.

The peer sends an authentication request to the other peer, performs ECC-based shared-secret validation, and retrieves the file securely. This interaction highlights a unique advantage of P2P networks: dynamic resource sharing. Instead of failing a request or depending on a central store, the peers cooperate autonomously to complete the user's request. This improves availability, reliability, and system robustness.

4. RESULTS AND ANALYSIS

This research demonstrates a complete end-to-end secure peer-to-peer (P2P) file-sharing architecture using a Django web framework integrated with socket-based peer communication. The implementation emphasizes secure file transfer, peer selection, computation-time measurement, and

performance visualization through dynamically generated graphs. The system coordinates user authentication, file operations, and cost comparison between existing and proposed algorithms. Below is a detailed explanation of the major modules and workflow based on the provided Python code.

1. Library Imports and Environment Setup

- Imports Python libraries for core backend operations including os, socket, pickle, io, base64, NumPy, and Django utilities.
- Uses Py MySQL to establish communication with the MySQL database for storing and retrieving user credentials.
- Imports matplotlib.pyplot to generate runtime computation graphs for comparing algorithm performance.
- Initializes global variables (existing_cost, propose_cost, uname) used for tracking user sessions and performance metrics.

2. User Authentication and Session Handling

- The User Login Action function validates user credentials by querying the MySQL database using Py MySQL.
- If authenticated, the username is stored in a global variable and the user is directed to the dashboard screen.
- The Register Action function inserts new user information into the database after checking for duplicates.
- Auxiliary views (User Login, Register, index) handle template rendering for the frontend.
- These modules ensure secure and controlled access to all file-sharing operations.

3. File Upload Module

- Implemented in Upload File Action, this module allows users to upload files to Peer1 or Peer2.
- The selected peer determines the port number (3333 for Peer1, 4444 for Peer2).
- The uploaded file is read as raw bytes, packaged into a Python list (["save", filename, filedata]), serialized using pickle.dumps(), and sent over a socket using sendall().
- The peer server acknowledges the operation by sending a confirmation message back to Django.
- The message is decoded and displayed on the User Screen, completing the upload workflow.

4. File Download Module

- The Download function initiates the download process by capturing computation start-time using timeit.default_timer().
- A socket connection is opened to the corresponding peer server based on user input.
- A download command is serialized with pickle and sent to the peer.
- The peer responds by sending raw file data back over the socket.
- The system computes propose_cost based on measured time and simulates existing_cost by multiplying the duration by 1.3.
- The downloaded file content is returned to the user as a forced-download HTTP response using Django's Http Response.

5. Dynamic File Listing

- The DownloadFile function scans both peer directories (Data/peer1, Data/peer2) using os.walk().
- All filenames are added as HTML <option> elements and passed to the frontend.
- This allows users to easily select files available in either peer node for download.

6. Performance Measurement

- During file download, computation time is recorded using Python's timing utilities.

- These recorded values represent the cost of the proposed and existing algorithms.
- Computation costs are stored in global variables for later visualization.

7. Graph Generation and Visualization

- The Graph function generates a bar chart comparing existing and proposed computation costs.
- Uses NumPy to create arrays for bar positions and labels.
- Matplotlib generates a 2D bar chart representing performance differences.
- The plot is saved into an in-memory buffer (Bytes IO), converted into Base64 encoding, and injected into the template directly.
- This eliminates the need for storing graph images on the server.

8. Socket-Based Peer Interaction

- All peer interactions occur over raw TCP sockets.
- The client sends structured commands (save/download) and receives acknowledgments or binary file data.
- Serialization using pickle simplifies message formation, while socket APIs ensure low-level control of data transmission.
- Peer nodes operate on fixed ports, making the architecture deterministic and easy to manage.

9. Template Rendering and UI Integration

- Django's template engine renders dynamic HTML pages (UploadFile.html, DownloadFile.html, UserScreen.html).
- Context variables (messages, filenames, graph images) are passed seamlessly to the UI.
- The interface guides users through uploading, downloading, graph viewing, and session actions

Fig. 3 illustrates the computation cost comparison between the existing method and the proposed lightweight cryptographic authentication model. The bar chart represents the time required for key agreement and secure file download operations. It clearly shows that the proposed algorithm requires less computation time compared to the existing approach. This reduction in computation cost demonstrates the efficiency of using elliptic curve–based authentication in the peer-to-peer file exchange system. The result confirms that the proposed model achieves improved performance while maintaining strong security.

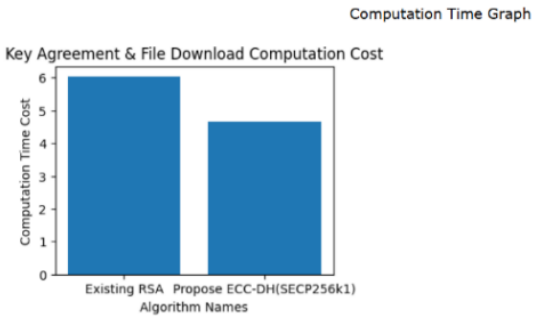

Fig. 3: Computational cost for proposed system.

5. CONCLUSION

The research demonstrates the development of a decentralized and secure file-sharing system by integrating a Django-based web application with a P2P network and ECC. It overcomes key drawbacks of traditional centralized systems, including single-point failure, identity exposure, and reliance on centralized authentication. The implementation of ECC-based key exchange ensures secure and anonymous communication between peers while maintaining data confidentiality and integrity. Lightweight cryptographic operations help in reducing

computational complexity without compromising security. The distributed architecture enables direct interaction between nodes, improving scalability and system robustness. The Django framework efficiently manages user authentication, database operations, file handling, and performance visualization. Experimental analysis indicates that the system achieves lower latency and reduced overhead compared to conventional approaches such as RSA-based models.

REFERENCE

[1] Cao, J.; Yan, Z.; Ma, R.; Zhang, Y.; Fu, Y.; Li, H. LSAA: A Lightweight and Secure Access Authentication Scheme for Both UE and mMTC Devices in 5G Networks. IEEE Internet Things J. 2020, 7, 5329–5344.

[2] Zhao, J.; Liu, J.; Yang, L.; Ai, B.; Shanjin, N. Future 5G-oriented system for urban rail transit: Opportunities and challenges. China Commun. 2021, 18, 1–12.

[3] Guo, F.; Yu, F.R.; Zhang, H.; Li, X.; Ji, H.; Leung, V.C.M. Enabling Massive IoT Toward 6G: A Comprehensive Survey. IEEE Internet Things J. 2021, 8, 11891–11915.

[4] Zhao, J.; Ni, S.; Yang, L.; Zhang, Z.; Gong, Y.; You, X. Multiband Cooperation for 5G HetNets: A Promising Network Paradigm. IEEE Veh. Technol. Mag. 2019, 14, 85–93.

[5] Jiang, Q.; Zhang, N.; Ni, J.; Ma, J.; Ma, X.; Choo, K.K.R. Unified Biometric Privacy Preserving Three-Factor Authentication and Key Agreement for Cloud-Assisted Autonomous Vehicles. IEEE Trans. Veh. Technol. 2020, 69, 9390–9401.

[6] Zhong, Hong; Zhang, Chuanwang; cui, Jie; Xu, Yan; Liu, Lu (2020). Authentication and Key Agreement Based on Anonymous Identity for Peer-to-Peer Cloud. University of Leicester. Journal contribution. https://hdl.handle.net/2381/12562322.v1

[7] Tyagi P, Kumari S, Alzahrani BA, Gupta A, Yang M-H. An Enhanced User Authentication and Key Agreement Scheme for Wireless Sensor Networks Tailored for IoT. *Sensors*. 2022; 22(22):8793. https://doi.org/10.3390/s22228793

[8] Chen Y, Martínez J-F, Castillejo P, López L. An Anonymous Authentication and Key Establish Scheme for Smart Grid: FAuth. *Energies*. 2017; 10(9):1354. https://doi.org/10.3390/en10091354

[9] Ali ZA, Abduljabbar ZA, AL-Asadi HAA, Nyangaresi VO, Abduljaleel IQ, Aldarwish AJY. A Provably Secure Anonymous Authentication Protocol for Consumer and Service Provider Information Transmissions in Smart Grids. *Cryptography*. 2024; 8(2):20. https://doi.org/10.3390/cryptography8020020

[10] Peivandizadeh A, Y. Adarbah H, Molavi B, Mohajerzadeh A, H. Al-Badi A. A Secure Key Exchange and Authentication Scheme for Securing Communications in the Internet of Things Environment. *Future Internet*. 2024; 16(10):357. https://doi.org/10.3390/fi16100357

[11] Zhao J, Hu H, Huang F, Guo Y, Liao L. Authentication Technology in Internet of Things and Privacy Security Issues in Typical Application Scenarios. *Electronics*. 2023; 12(8):1812. https://doi.org/10.3390/electronics12081812

[12] Wu T-Y, Wang L, Chen C-M. Enhancing the Security: A Lightweight Authentication and Key Agreement Protocol for Smart Medical Services in the IoHT. *Mathematics*. 2023; 11(17):3701. https://doi.org/10.3390/math11173701

[13] Deep G, Mohana R, Nayyar A, Sanjeevikumar P, Hossain E. Authentication Protocol for Cloud Databases Using Blockchain Mechanism. *Sensors*. 2019; 19(20):4444. https://doi.org/10.3390/s19204444

[14] Lee J, Kim M, Park K, Noh S, Bisht A, Das AK, Park Y. Blockchain-Based Data Access Control and Key Agreement System in IoT Environment. *Sensors*. 2023; 23(11):5173. https://doi.org/10.3390/s23115173

[15] Kairaldeen AR, Abdullah NF, Abu-Samah A, Nordin R. Peer-to-Peer User Identity Verification Time Optimization in IoT Blockchain Network. *Sensors*. 2023; 23(4):2106. https://doi.org/10.3390/s23042106