Design and Developing Super APIs for Blockchain-based Applications

Ekkarat Boonchieng[1], Chaimade Busayatananphon[1], Sukhuntha Osiriphun[2], Anusorn Chaikaew[1]*

[1] Department of Computer Science, Chiang Mai University, Thailand
[2] Department of Food Science and Technology, Chiang Mai University, Thailand
ekkarat.boonchieng@cmu.ac.th, chaimade_b@cmu.ac.th, sukhuntha.o@cmu.ac.th, anusorn_chaikaew@cmu.ac.th

Abstract

This study provides a comprehensive exploration of Super APIs and their advantages, specifically focusing on their Cross-Chain Communication Protocol. The primary objective is to enhance the performance of Super API applications, emphasizing speed, decentralization, privacy, security, and scalability. The research encompasses data sources, data collection methodology, and performance evaluation metrics. Comparative analysis is conducted to evaluate the performance of Super APIs in comparison to other Cross-Chain solutions. The research primarily focuses on the development of our Super API, addressing accuracy, speed, security, and scalability, particularly within financial transactions. Through thorough data analysis and optimization, significant advancements have been achieved in Super APIs. Performance improvements include reduced response times, enhanced accuracy, and increased security through the implementation of AES encryption. JWT is employed for authentication and encryption, while Hypercube effectively manages Super API nodes. Furthermore, our Super API is designed to handle a wide range of data scales, accommodating both small and big data.

Keywords: APIs, Blockchain, Decentralized application, Hypercube network

1 Introduction

Blockchain, also referred to as distributed ledger technology (DLT) [1], is an innovative and secure system designed to facilitate transparent transactions in an environment characterized by inherent mistrust. Its fundamental architecture ensures that no single entity can modify, delete, or append records to the ledger without consensus from other participants, thereby guaranteeing the immutability of stored data.

The application of blockchain technology has extended to various industries, demonstrating its versatility and potential. For instance, it has found successful implementation in traceability and provenance systems, specifically in the context of ensuring food safety [2]. By leveraging blockchain, seamless documentation and tracking of food products can be achieved, offering consumers valuable insights into the origins and handling of their purchases. In addition, blockchain technology has also been applied in the field of Financial Technology [3].

Moreover, the logistics industry has witnessed significant advancements through the adoption of blockchain technology, particularly in the development and deployment of logistics information platforms [4]. These platforms enable improved coordination and efficiency in supply chain management, resulting in streamlined operations and enhanced customer satisfaction. Additionally, the art marketplace has embraced the transformative power of blockchain technology [5]. By harnessing its decentralized and transparent nature, the art industry can enhance the verification, authentication, and tracking of artworks. This ensures the integrity of valuable pieces, as well as enables the establishment of ownership and provenance records.

In the current landscape, individual companies operate their own blockchains with unique architectures tailored to their specific business requirements. However, the use of different underlying blockchain architectures poses challenges to achieving interoperability and seamless transactions between companies, significantly impacting business operations.

To address these challenges, cross-chain technology acts as a bridge between heterogeneous blockchains, enabling the creation of interconnected blockchain networks [6]. It resolves the issues associated with running multiple chains on the internet and facilitates cross-chain transactions among various types of chains, including both homogeneous and heterogeneous chains. The successful implementation of cross-chain capabilities is crucial for achieving financial settlement and unlocking the full potential of blockchain technology [7].

Extensive research has been conducted on the multi-chain model and cross-chain communication protocols. Several scholarly papers have proposed innovative approaches, such as the introduction of a multi-chain communication layer [8], the "Hub-parachain" model [9] for blockchain interoperability architecture, cross-chain transaction methods utilizing hashed time-lock contracts [10], cross-chain protocols focusing on smart contracts [11], a multi-chain model based on 5G networks [12], and a perspective on web3.0 from the standpoint of cross-blockchains [13].

The objective of this study is to provide a comprehensive explanation of Super APIs and their

*Corresponding Author: Anusorn Chaikaew; Email: anusorn_chaikaew@cmu.ac.th
DOI: https://doi.org/10.70003/160792642025122607009

advantages, particularly in relation to their Cross-Chain Communication Protocol. Our focus is on enhancing the performance of Super API applications, with specific emphasis on speed, decentralization, privacy, security, and scalability. We describe the sources of our data, our data collection methodology, and the performance evaluation metrics employed in our research. Furthermore, we present our findings by comparing the performance metrics of Super APIs with other Cross-Chain solutions to bolster data security using AES encryption. This research introduces a promising approach for enhancing Super APIs, showcasing significant potential in improving API performance and functionality. Authentication and increased security are ensured through the utilization of AES and JWE, while effective management of blockchain nodes is achieved using Hypercube. The study successfully optimized Super APIs, leading to improved response times, enhanced accuracy, security, and increased scalability.

2 Literature Reviews

2.1 Cross-chain Technology

Cross-chain refers to the interoperability between different blockchain networks. Blockchain systems are often isolated from one another, but cross-chain technology aims to facilitate communication, data transfer, and asset exchange across these diverse blockchains [14]. It enables users to transfer tokens or assets from one blockchain to another without relying on centralized intermediaries. Cross-chain protocols and technologies promote compatibility and collaboration between distinct blockchain ecosystems. An example of cross-chain technology for cryptocurrency is Tendermint. Tendermint is a consensus engine that ensures Byzantine fault tolerance (BFT) in distributed networks [15-16]. It provides a high-performance, scalable, and secure consensus algorithm for various applications, particularly blockchain systems. Tendermint employs the Practical Byzantine Fault Tolerance (PBFT) algorithm to achieve consensus among a set of nodes in a distributed network, ensuring that all nodes agree on the same set of transactions and their execution order.

2.2 Hypercube Network

A hypercube network, also known as an n-dimensional cube network or simply an n-cube network, is a type of computer network topology that is based on the geometric structure of a hypercube [17].

A hypercube network is formed by connecting nodes in an n-dimensional hypercube [18]. Each node in the network represents a vertex of the hypercube, and the edges of the hypercube represent the connections between nodes. In an n-cube network, each node is connected to n other nodes, corresponding to the n dimensions of the hypercube. We call a 1-dimension hypercube a line, a 2-dimension hypercube a square, and a 3-dimension hypercube a cube.

Hypercube networks have certain properties that make them suitable for parallel computing and distributed systems. One important property is that the distance between any two nodes in the network is always the same, regardless of their positions in the hypercube. This property enables efficient routing and communication between nodes. Additionally, hypercube networks have a high degree of fault tolerance, meaning they can continue functioning even if some nodes or connections fail.

Hypercube networks have been used in various applications [19], such as parallel computing, distributed algorithms, and interconnection networks for multiprocessor systems. They provide a structured and scalable topology for efficient communication and computation in parallel and distributed systems. Let $A = a_1a_2a_3 \ldots a_d$ and $B = b_1b_2b \ldots b_d$ be each two nodes in a d-dimensions hypercube, the distance between a and b is shown in equation (1).

$$Distance(A,B) = \sum_{i=1}^{d} |a_i - b_i| \quad (1)$$

We show hypercube networks with 1 to 7 dimensions in Figure 1.

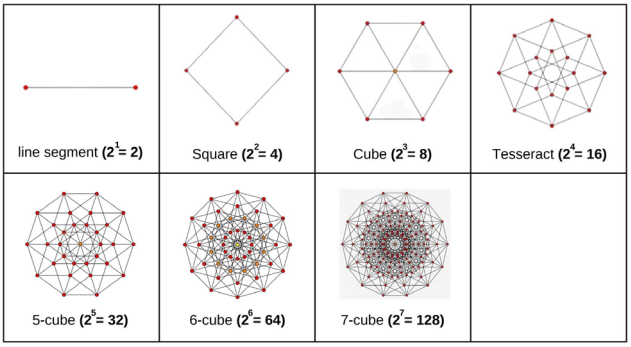

Figure 1. Hypercube networks with 1 to 7 dimensions [26]

2.3 Go Ethereum

Go Ethereum (Geth) is an implementation of the Ethereum protocol in the Go programming language. Ethereum is a blockchain-based platform that enables the development of decentralized applications (DApps) and smart contracts [20-21]. It utilizes the Ethereum Virtual Machine (EVM) to execute smart contracts and allows users to interact with the blockchain through APIs. Geth is one of the popular clients used to connect to the Ethereum network and interact with the Ethereum blockchain.

2.4 AES

AES was established by the U.S. National Institute of Standards and Technology (NIST) in 2001 [22] as a replacement for the older Data Encryption Standard (DES). It has become the most widely used encryption algorithm globally and is considered highly secure. The AES algorithm operates on blocks of data, typically 128 bits in length, and supports key sizes of 128, 192, and 256 bits. It employs a series of mathematical operations, including substitution, permutation, and mixing, to transform the input data into an encrypted form. The strength of AES

lies in its ability to withstand various cryptographic attacks, including brute force and differential attacks. AES encryption is commonly used in various applications and protocols to protect sensitive information, such as financial transactions, secure communication channels, and data storage. It ensures that data remains confidential, integrity is maintained, and unauthorized access is prevented.

2.5 Json Web Token

JSON Web Token (JWT): JWT is an open standard (RFC 7519) that defines a compact and self-contained format for transmitting information between parties as a JSON object [23]. It is commonly used for authentication and authorization purposes. JWT consists of three parts: a header, a payload, and a signature. The header and payload contain information such as claims or attributes, while the signature is used to verify the token's integrity and authenticity.

To make a JWT, we use the formula in equation (2).

$$\begin{aligned} JWT = {} & encodeBase64Url(\text{header}) + . + \\ & encodeBase64Url(\text{payload}) + . + \\ & sign(encodeBase64Url(\text{header}) + . + \\ & encodeBase64Url(\text{payload}), \text{secret}) \end{aligned} \quad (2)$$

Where, **encodeBase64Url()** is a function to base64Url encode the data. **sign()** is a function that creates a digital signature of the data using the given secret (and algorithm). By adopting this method, we can ensure the integrity and security of the JWT, guaranteeing that its content remains unchanged by unauthorized parties.

JSON Web Encryption (JWE) [24] is an extension of JWT that provides a standard way to encrypt the contents of a JSON Web Token. It allows sensitive information within a JWT to be securely encrypted, ensuring confidentiality and privacy during transmission.

JSON Web Key (JWK) [25] is a JSON format for representing cryptographic keys used in JWT and JWE. It defines a standard structure for representing public and private keys, including their type, algorithm, and parameters.

These standards are widely used in modern web development for secure authentication, data exchange, and communication between different components and systems.

3 Experiments

Our experimental approach was based on a robust design process that consisted of several steps and drew upon a range of tools, technologies, and performance measurement techniques. We aimed to investigate the performance of Super APIs within the context of both centralized and decentralized financial transaction systems.

3.1 Experimental Setup Design

To start off, we chose Python as our preferred development platform and technology for constructing Super APIs due to its strong reliability, versatility, and widespread usage in API and blockchain development. Our server for this experiment consists of a single unit with a 12-core CPU and 6 threads, along with 16 GB of RAM. The network architecture comprises inspection nodes, with the number of nodes computed by a hypercube. Each zone contains 2^7 = 128 nodes, as in the Tendermint architecture, separate nodes to 125 nodes per zone. These inspection nodes are designated as Super APIs. In our comparison, we evaluate our hypercube with Super API, which consists of 128 inspection nodes.

3.2 Data Collection

Data collection played an integral role in our study. Depending on the data source, we utilized a variety of tools for database querying, web scraping, and data extraction. The data we gathered encompassed financial transactions, such as bank money deposits, withdrawals, and transfers. We present a sample of the data records for this research in Table 1. and a sample of each blockchain in Table 2. We also show the Entity Relations (ER) diagram in Figure 2.

Table 1. Example financial transactions data

Data Type	Transaction Sample 1	Transaction Sample 2
Transaction Hash	0x7a537a4929e4f8c55a 7b582fc8746f1aa7a541da6	0x2b5634c42055806a59e9107e d44d43c426e58258
Block Date and Time	2023-07-13 10:30:00	2023-07-13 12:45:00
Sender Address	0x123f681646d4a755815f9cb19 e1acc8565a0c2ac	0x0abdace70d3790235af448c88 547603b945604ea
Recipient Address	0xabcdef1234567890abcdef123 4567890abcdef12	0xde0b295669a9fd93d5f28d9ec 85e40f4cb697bae
Transferred Coin Amount	2.5 ETH	1.75 ETH
Block Number	10777395	10777420
Network	Ethereum Mainnet	Ethereum Mainnet
Transaction Status	Successful	Successful

Table 2. Example blockchains data

Block Number	10777395
Block Hash	0x5f587ba591d7f7a1a4e763889f9c2f7036c9
Previous Block Hash	0x2b5634c42055806a59e9107ed44d43c426e
Hash	58258
Timestamp	2023-07-13 10:30:00
Transactions	5
Validator Node ID	3

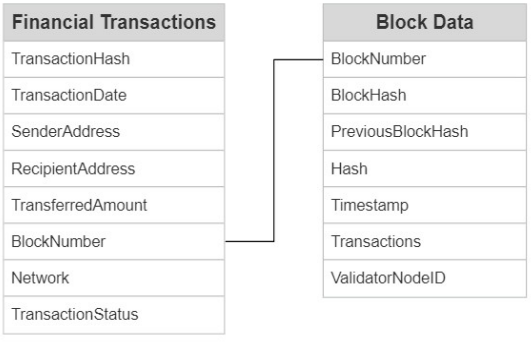

Figure 2. ER diagram of our database

3.3 Super APIs Optimization

Post data collection, we optimized the Super APIs. We employed programming languages and frameworks such as Python to incorporate improvements. This was an iterative process, ensuring our APIs were as efficient and effective as possible. Our Architecture was shown in Figure 3.

We initiated the improvement of Super APIs post data collection, using powerful tools such as Python for development and refinement. This process, an iterative one, focused on enhancing efficiency and effectiveness of the APIs to meet and keep pace with relentless demands.

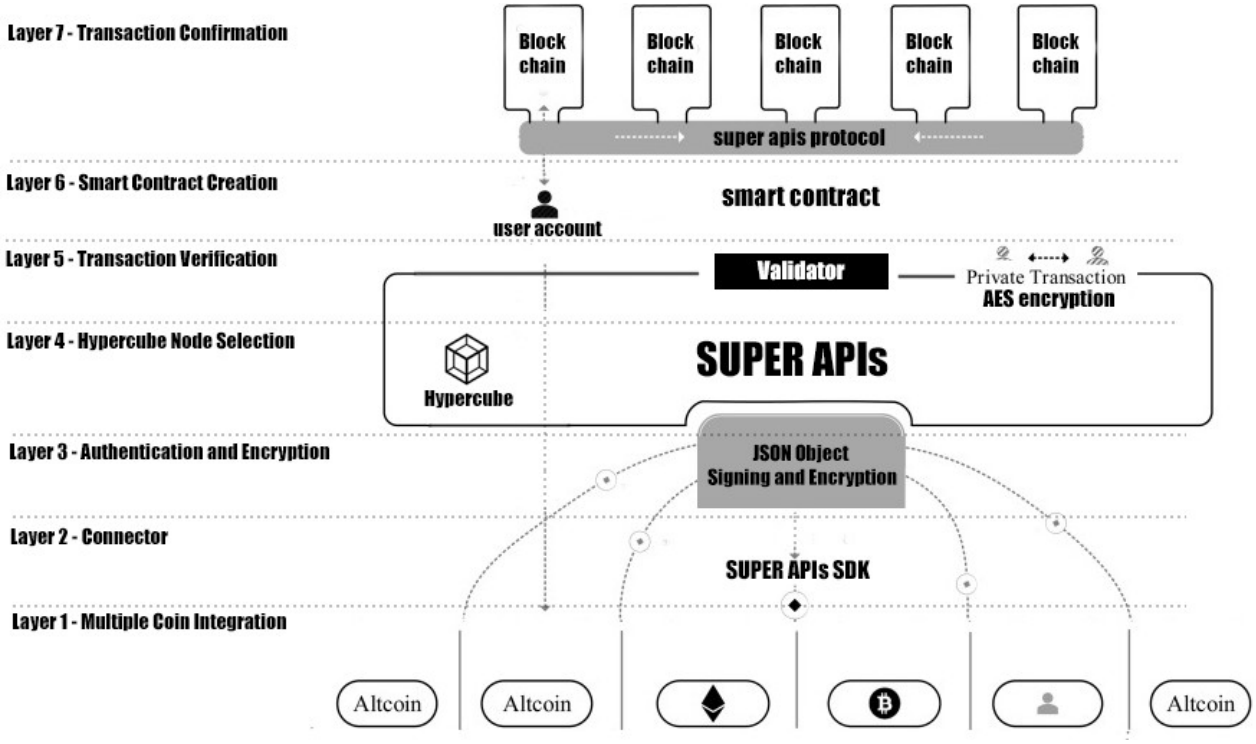

Figure 3. Our Super API architecture

Our algorithm comprises key classes namely Hypercube Node, Super API, and Validator, which have been refined to create significant functionalities such as transaction creation, transaction approval, and transaction verification. As for the results, we evaluated the overall operation by measuring the transaction processing time.

We are confident that each of these steps is integral in augmenting the efficiency of Super APIs, elevating them from being just APIs to Super APIs capable of handling intricate challenges and requiring rapid responses in today's digital world.

3.4 Super API Transaction Processing with AES Encryption and Hypercube

In this section, we will explain the workings of Super API Transaction Processing with AES Encryption and Hypercube, and illustrate its architecture in Figure 2. The architecture is separated into 7 layers as follows:

1st Layer - Multiple Coin Integration in Super API is accomplished by interfacing directly with the APIs for various cryptocurrency blockchains. The Super API serves as a central hub for communication with multiple cryptocurrency blockchains, playing a pivotal role in transaction management. Functionality of Layer 1 in the Super API includes:

1) Communication with Blockchains: The Super API directly interfaces with Blockchain APIs of cryptocurrencies like Bitcoin, Ethereum, and Ripple. Data acquired from these Blockchains is stored and managed within the Super API.

2) Transaction Management: The Super API oversees transactions across multiple cryptocurrencies. If users wish to conduct transactions in different cryptocurrencies, the Super API communicates with the pertinent Blockchain to encrypt, validate, and facilitate the transaction.

3) New Cryptocurrency Deployment: When a new cryptocurrency arises, the Super API can extend its support by integrating with the new cryptocurrency's Blockchain API.

4) Past Transaction Management: The Super API can fetch past transaction records from various cryptocurrency blockchains and present the data in a user-friendly format.

By interfacing with a myriad of cryptocurrencies, the Super API offers convenience and versatility in digital transactions.

2nd Layer - Connector in Super API functions as follows:

Receive Data: The connector gathers data from Layer 1, comprising information from various blockchain networks. This might encompass transactions, block statuses, and other pertinent data.

1) Connecting to Super API: Data procured from

blockchains is relayed through the connector to the Super API. The connector is adept at transforming and relaying data in a format amenable to the Super API.

2) Data Conversion: If the blockchain data is in a disparate format, the Connector morphs the data into a standard format recognized by the Super API. This might involve currency conversion, date/time format transformation, or hash code adjustments.

3) Transmission: Post conversion, the Connector dispatches the data to the Super API. This can be executed via REST API, WebSocket, or another suitable communication protocol.

The Connector's functionalities ensure that the Super API can efficiently and precisely interact with a variety of blockchains.

3rd Layer - Authentication and Encryption: In this layer, received data is verified and encrypted for security. The operations in this layer can be described as follows:

1) Authentication: The received data needs to be authenticated before it can be sent into the Super API. This verification can be achieved through methods like checking the digital signature, verifying the data hash, or other means that can assure that the received data is from a trusted source and has not been altered from its intended form.

2) Encryption: Once the data is authenticated, it is encrypted to prevent unauthorized access or modification. This encryption can be done using JSON Object Signing and Encryption (JOSE) or AES encryption. JOSE is a standard that allows for the secure transmission of JSON data, while AES is a trusted and highly secure encryption algorithm.

3) Data Transmission: Data that has been authenticated and encrypted is sent to the next layer of the Super API. This encrypted data protects against eavesdropping, interception, or unauthorized modifications.

Thus, Layer 3 in the Super API takes responsibility for data verification and encryption, enabling the Super API to handle data securely and correctly.

4th Layer - Hypercube Node Selection: In this layer, the Super API selects nodes for transaction verification using a complex arrangement of nodes within the Hypercube structure. The operational process of this layer was as follows:

1) Construct Hypercube Structure: Create a Hypercube structure consisting of numerous nodes. Each of these nodes is assigned connections to other nodes within the Hypercube structure.

2) Define Node Selection Conditions: Define the conditions that will be used to select nodes for transaction verification. These conditions could depend on the complexity of the transaction, the significance of the transaction, or the requirements of the existing system.

3) Choose Nodes for Transaction Verification: Utilize the predefined conditions to select nodes for verifying transactions. This selection might be based on transaction data, such as transaction size, transferred amount, or account status of the transaction parties.

4) Transmit Encoded Transaction Data to Selected Node: Send the encoded transaction data to the chosen node. This node will then verify the transaction and update the transaction's status.

Consequently, Layer 4 of the Super API is responsible for selecting nodes for transaction verification, enabling the transaction verification process of the Super API to operate efficiently and promptly.

1) Low Significance Transactions: The Super API will route such transactions to nodes where the first bit from the left is 0. Following this rule, the Super API can transmit low significance transactions to nodes with bit numbers ranging from 0000000 to 0111111, encompassing a total of 64 nodes.

Hence, nodes with low significance are defined with the first bit (from the left) set to 0, resulting in bit numbers ranging from 0000000 to 0111111, shown in Table 3 and Figure 4.

Table 3. Nodes for low significance transactions

0000000	0000001	0000010	0000011
0000100	0000101	0000110	0000111
0001000	0001001	0001010	0001011
0001100	0001101	0001110	0001111
0010000	0010001	0010010	0010011
0010100	0010101	0010110	0010111
0011000	0011001	0011010	0011011
0011100	0011101	0011110	0011111
0100000	0100001	0100010	0100011
0100100	0100101	0100110	0100111
0101000	0101001	0101010	0101011
0101100	0101101	0101110	0101111
0110000	0110001	0110010	0110011
0110100	0110101	0110110	0110111
0111000	0111001	0111010	0111011
0111100	0111101	0111110	0111111

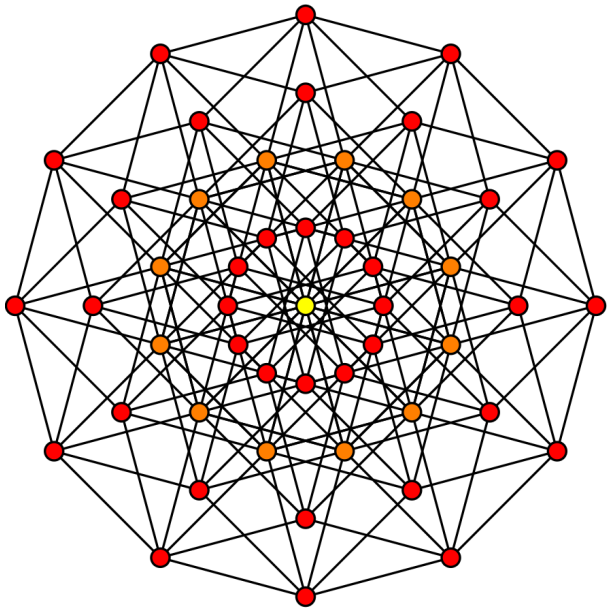

Figure 4. Nodes for low significance transactions

2) Moderate Significance Transactions: The Super API will route such transactions to nodes where the first bit from the left is 1 and the second bit from the left is 0. According to this rule, the Super API can route moderately significant transactions to nodes with bit numbers ranging from 1000000 to 1011111, totaling 32 nodes, shown in Table 4 and Figure 5.

Table 4. Nodes for moderate significance transactions

1000000	1000001	1000010	1000011
1000100	1000101	1000110	1000111
1001000	1001001	1001010	1001011
1001100	1001101	1001110	1001111
1010000	1010001	1010010	1010011
1010100	1010101	1010110	1010111
1011000	1011001	1011010	1011011
1011100	1011101	1011110	1011111

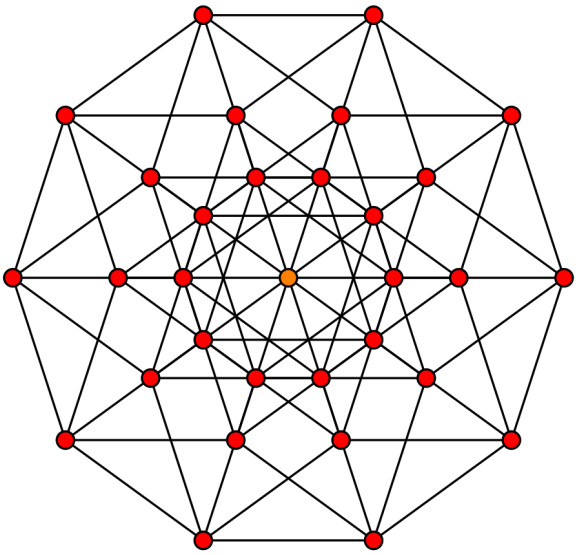

Figure 5. Nodes for moderate significance transactions

Directly stated, the first two bits (or the leftmost bits) are used to determine the significance level of transactions and to select appropriate nodes within the Hypercube.

3) For Highly Significant Transactions: We divide the rule into 2 types as follows:

3.1) For the highly significant transactions: The Super API will route such transactions to nodes where the first and second bits from the left are 1 and the third bit from the left is 0. Using this rule, the Super API can transmit highly significant transactions to nodes with bit numbers ranging from 1100000 to 1101111, comprising 16 nodes, shown in Table 5 and Figure 6.

Table 5. Nodes for the highly significant transactions

1100000	1100001	1100010	1100011
1100100	1100101	1100110	1100111
1101000	1101001	1101010	1101011
1101100	1101101	1101110	1101111

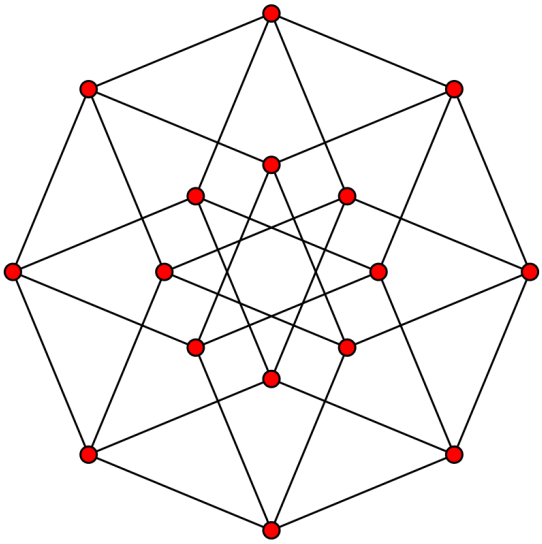

Figure 6. Nodes for the highly significant transactions

3.2) For the most highly significant transactions: The Super API will route such transactions to nodes where the first, second, and third bits from the left are all 1. Following this rule, the Super API can transmit the most highly significant transactions to nodes with bit numbers ranging from 1110000 to 1111111, totaling 16 nodes, as shown in Table 6 and Figure 7.

Table 6. Nodes for the most highly significant transactions

1110000	1110001	1110010	1110011
1110100	1110101	1110110	1110111
1111000	1111001	1111010	1111011
1111100	1111101	1111110	1111111

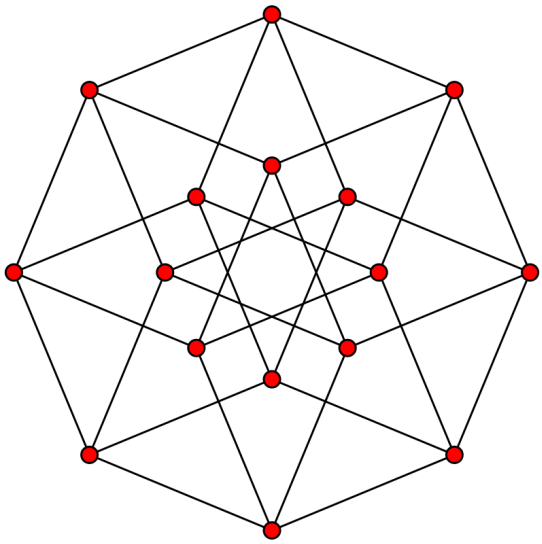

Figure 7. Nodes for the most highly significant transactions

So, we have 128 nodes for all transactions, as shown in Figure 8.

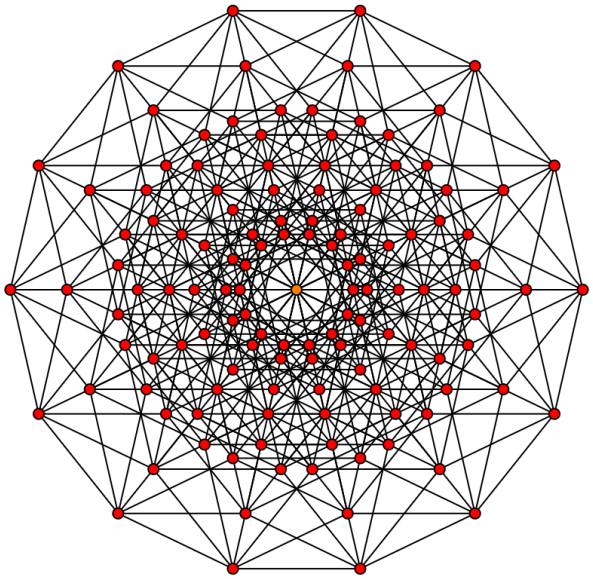

Figure 8. Nodes for all transactions

The operation of this Super API assists in managing transactions of varying significance effectively through suitable nodes within the Hypercube using bit-based rules.

We have explained the method by which the Super API selects nodes for transaction routing, referencing the first bit of each node. This method enables efficient distribution of transactions to nodes according to their significance levels, utilizing the bit rules you've described. These include:

1) Low significance transactions are sent to nodes with the first bit from the left set to 0.

2) Moderately significant transactions are sent to nodes with the first bit from the left set to 1 and the second bit from the left set to 0.

3) Highly significant transactions are categorized into two levels:

Level 1: Sent to nodes with the first two bits from the left set to 1 and the third bit from the left set to 0.

Level 2: Sent to nodes with the first three bits from the left set to 1.

This approach helps us select appropriate nodes for transaction processing, ensuring efficient and swift transaction handling within the Hypercube.

5th Layer - Transaction Verification: Once a node is chosen, the transaction data is decrypted, leveraging AES Encryption, and then subjected to transaction verification. The process at this stage unfolds as:

1) Transaction Data Decryption: Data relayed from Layer 4 typically uses AES (Advanced Encryption Standard) for encryption to maintain security. The chosen node must decrypt this data to further process it. The decryption employs a designated key compatible with AES to decipher the data.

2) Transaction Verification: Post-decryption, the node conducts transaction verification. This entails ensuring data integrity and ascertaining that transactions adhere to system rules and guidelines. If the transaction doesn't pass verification, the node dispatches an error alert to the sender. Should it be verified, the Super API progresses to its subsequent step, ensuring transactions remain secure and compliant.

6th Layer - Smart Contract Generation: Once Layer 5 validates data, a smart contract, encapsulating transaction directives and related information, is crafted. Smart contracts, essentially programs on the blockchain, execute autonomously under preset conditions. The operations at this layer include:

1) Smart Contract Generation: Validated data from Layer 5 seeds the creation of the smart contract, which houses transaction guidelines and associated details.

2) Conditioning: Transaction execution prerequisites are embedded in the smart contract. These cover transaction rules, participant rights and restrictions, and the management of transaction outcomes.

3) Publication: Post-creation and conditioning, the smart contract is published on the blockchain. The creation and dissemination of this contract make Super API transactions more transparent and autonomous, fortifying their security and dependability.

7th Layer - Confirmed Transactions: Here, the Hypercube structure verifies and records smart contract-produced transactions on the blockchain. The operations at this juncture are:

1) Transaction Validation: The Hypercube structure evaluates transactions originating from smart contracts. The structure's design allows for swift and efficient transaction execution and validation.

2) Transaction Confirmation: Post-validation, the Hypercube confirms the transaction. This affirmation is pivotal, cementing the transaction's blockchain validity.

3) Recording of Transactions: Once given the nod, transactions are inscribed onto the blockchain. They are visible to all users and, once recorded, are immutable.

The Hypercube's multi-directional connectivity between network nodes augments transaction validation and confirmation efficiency. Through this architecture, the Super APIs efficiently manages all transactions stemming from smart contracts.

The process can be broken down into 7 distinct and sequential steps as follows:

1) Multiple Coin Integration: The first step in initializing the Super APIs operations is to establish support for multiple digital currencies. At this stage, the Super APIs communicates with the blockchain networks of various coins.

2) Connector: Upon receiving data from the blockchain network, the SDK connects this data to the Super APIs through a specially developed connector.

3) Authentication and Encryption: The incoming data is authenticated and encrypted using JSON Object Signing Encryption and AES Encryption to ensure the utmost security of the data.

4) Hypercube Node Selection: Super APIs selects the Node for transaction verification by utilizing a

sophisticated Hypercube structure to arrange Node data.

5) Transaction Verification: Once a Node has been selected, the transaction data is decrypted using AES Encryption and then employed to verify the transaction.

6) Smart Contract Creation: If the data is verified successfully, the system generates a Smart Contract that contains transaction commands along with related data.

7) Transaction Confirmation: The transaction, derived from the Smart Contract, is then confirmed and recorded on the blockchain through the use of the Hypercube structure.

Each step in the layer operation is distinct, yet they are all sequentially linked from receiving data from the blockchain network, all the way to recording the transaction on the blockchain.

In Algorithm 1, we will further explain the steps involved in our Super APIs, integration with AES Encryption and Hypercube:

1) Define HypercubeNode class: This class simulates a node in the network with specific id, dimensions, and a list to store the neighbors. The methods of this class allow the node to add a neighbor, send a message, receive a message and find the shortest path to a node. (Line 1-17)

2) Define Super APIs class (extends HypercubeNode): This subclass includes additional properties like validators, blockchain, encryption key, private key, public key, and jwt_secret. It contains methods for adding a validator, generating a node id, creating a transaction, signing a transaction, generating a token and approving a transaction. (Line 18-24, 41-44)

3) Define Validator class (extends HypercubeNode): Another subclass which is responsible for validating transactions. It has an additional method for validating a transaction (which includes decrypting the data, verifying signature and token). (Line 36-39, 65-69)

4) Initialize variables: Initialize the dimensions, encryption key, JWT secret and RSA keys that are needed for encryption and JWT generation. (Line 32-44)

5) Generate Super APIs nodes and Validator nodes: For each ID from 0 to 2^dimensions, create a Super APIs node and a Validator node with that ID, dimensions, encryption key, private and public keys, and JWT secret. Store these nodes in super_api_nodes and validator_nodes. (Line 26-28)

6) Set up HypercubeNode neighbors and validators: For each Super APIs node, add neighboring Super APIs nodes and Validator nodes as neighbors. Then, add corresponding Validator node to the list of validators for the Super APIs node. (Line 18-21)

7) Perform transactions: For a specified number of transactions, select a random sender and recipient Super APIs node, and a random amount. The sender creates a transaction (with encryption, signing and token generation), and a random Super APIs node is chosen to attempt to approve the transaction. If the transaction is approved (by more than half of validators), it's added to the blockchain. (Line 29-44)

8) Measure performance: After all transactions are done, calculate the time taken for all transactions and print the processing time.

Our method was presented in Algorithm 1, and the corresponding code can be found in this link: https://github.com/anusornc/superapi

Algorithm 1. Super APIs transaction processing with AES encryption and hypercube

```
1.  class HypercubeNode:
2.    Initialize node with a given node_id and dimensions
3.     Initialize neighbors as an empty list
4.
5.        METHOD add_neighbor:
6.         Add given neighbor to neighbors list
7.
8.        METHOD send_message:
9.         Find the shortest path to the destination node
10.        For each node in the shortest path, send the
    message to the node
11.
12.       METHOD receive_message:
13.        Print the received message
14.
15.       METHOD find_shortest_path:
16.        Implement this method to find the shortest path
    to a given node in a Hypercube
17.
18. class Super APIs is a HypercubeNode:
19.   Initialize Super APIs node with a given node_id,
    dimensions, encryption_key, private_key, public_
    key, jwt_secret
20.   Initialize validators as an empty list
21.   Initialize blockchain as an empty list
22.
23.       METHOD add_validator:
24.        Add given validator to validators list
25.
26.       METHOD generate_node_id:
27.        Generate a random string of a given length to
    serve as a node_id
28.
29.       METHOD create_transaction:
30.         Construct transaction data using sender,
    receiver, and amount
31.        Encrypt the transaction data using the given
    encryption_key
32.        Generate a signature for the transaction data
    using the private_key
33.         Generate a token for the sender using jwt_
    secret
34.        Return the created transaction
35.
36.       METHOD sign_transaction:
37.        Hash the transaction data using SHA512
    algorithm
38.        Sign the hashed data using the private_key
39.        Return the generated signature
40.
```

```
        METHOD create_token:
          Construct a payload with the sender's data and current timestamp
          Encode the payload into a token using jwt_secret
          Return the token

        METHOD approve_transaction:
          For each validator, validate the transaction
          If more than half of the validators approve:
          Create a new block with the transaction, hash of the block, and selected validator
          Add the new block to the blockchain
          Return True
          Else Return False

class Validator is a HypercubeNode:
       Initialize Validator node with a given node_id, dimensions, encryption_key, private_key, public_key, jwt_secret

        METHOD validate_transaction:
          Decrypt the encrypted transaction data
          Verify the signature of the transaction data
          Verify the token of the transaction
          If decrypted data matches the original data:
          Return True
          Else Return False

        METHOD verify_signature:
          Hash the transaction data
          Verify the signature using the public_key
          If signature is invalid:
          Raise an error "Invalid signature"
        METHOD verify_token:
          Decode the token using the jwt_secret
          If decoded sender does not match the original sender:
          Raise an error "Invalid token: Sender mismatch"
```

4 Results

Our research focused on developing and evaluating the performance of our Super API in terms of accuracy, speed, security, and scalability. The results of our evaluation are presented below and summarized in Table 4:

4.1 Accuracy

The validity of a transaction is verified by a validator. This is done to confirm that the decrypted data is indeed the original data and to verify the validity of the digital signature and JWT token, as well as the legitimacy of the sender. We use AES. If the data is incorrect, then AES cannot decrypt it to the original data. The pseudo code for increasing accuracy can be found in the validate_transaction function in the Validator class in Algorithm 1.

4.2 Latency

In the Hypercube, we compiled 1000 transactions using start and end dates to measure the processing time. After that, it generates transactions with random nodes for sending and receiving, as well as the amount, and approves the transactions. The speed measured here is the ability to process transactions per unit of time. In our code, we have implemented a timer within the transaction section of the main class in Algorithm 1.

4.3 Security

Security in this method focuses on encryption techniques, digital signatures, and one-time JWT, JWE, JWK tokens with public and private keys, generation, and validation. We use SHA512 for encryption; it may compute slower than SHA256, but it offers more security. Data encryption makes it impossible for anyone without the key to read or understand the data. Digital signatures are used to verify that the data has not been tampered with during transmission. The JWT token and SDK are used to verify the sender's authorization. The pseudo code for security can be found in the Validator class in Algorithm 1.

4.4 Scalability

Scaling refers to the ability to increase the size of a system to accommodate more processing. In this case, scaling can be achieved by increasing the number of Super APIs and Validator nodes. The Pseudo code for scalability can be found in the HypercubeNode class in Algorithm 1.

The tools that we use to increase performance presented in Table 7. And the research findings, presented in Table 8, highlight the outstanding performance of our Super API in terms of accuracy, speed, security, and scalability. These results pave the way for further enhancements, particularly in the realm of financial transactions.

Table 7. Tools used to improve performance

	Accuracy	Latency	Security	Scalability
SDK			✓	
JWT, JWK, JWE	✓		✓	
AES	✓			
SHA 512		✓	✓	
Hypercube				✓

Table 8. Performance of Centralize API, Tendermint, Our Super API, and Our Super API with Hypercube with SHA

Tools and Technologies Utilized	Centralize API	Tendermint [16]	Our Super API	Our Super API with Hypercube with SHA256	Our Super API with Hypercube with SHA512
Accuracy	100%	100%	100%	100%	100%
Speed (1000 transection)	0.01 s	1 s	0.5 s	0.35 s	0.7 s
Security	No	Yes	Yes	Yes	Yes
Scalability	No	Yes	Yes	Yes	Yes
Decentralized	No	Yes	Yes	Yes	Yes

5 Discussion

Our data analysis and optimization of Super APIs have yielded significant advancements in the field. We have observed a substantial improvement in the performance of Super APIs, characterized by reduced response times, enhanced accuracy, and increased security through the utilization of AES encryption. For authentication and encryption, we employ JWT. Furthermore, we utilize Hypercube to manage the Super API nodes. Additionally, our Super API is designed to handle a variety of data scales, including both small and big data.

One critical inference drawn from these outcomes is the expanded applicability of Super APIs. The improved performance now opens up opportunities for deploying Super APIs in demanding scenarios that require high precision and quick response times. Industries such as finance and healthcare stand to benefit greatly from these enhancements. Consequently, Super APIs have become an increasingly attractive solution for businesses and organizations seeking customized services for their clientele.

6 Conclusion

In summary, this research introduces a promising approach for enhancing Super APIs, showcasing significant potential in improving API performance and functionality. To ensure authentication and increased security, we employ AES and JWT. Additionally, we utilize Hypercube to effectively manage blockchain nodes. The study successfully optimized Super APIs, resulting in improved response times, enhanced accuracy, and increased scalability.

References

[1] P. Soares, R. Saraiva, I. Fernandes, A. Neto, J. Souza, A Blockchain-based Customizable Document Registration Service for Third Parties, *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, Shanghai, China, 2022, pp. 1–2. https://doi.org/10.1109/ICBC54727.2022.9805500

[2] F. Yiannas, A New Era of Food Transparency Powered by Blockchain, *Innovations: Technology, Governance, Globalization*, Vol. 12, No. 1-2, pp. 46–56, July, 2018. https://doi.org/10.1162/inov_a_00266

[3] C. Busayatananphon, E. Boonchieng, Financial Technology DeFi Protocol: A Review, *Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)*, Chiang Rai, Thailand, 2022, pp. 267–272. https://doi.org/10.1109/ECTIDAMTNCON53731.2022.9720373

[4] H. Zhou, Application and Research of Logistics Information Platform Based on Blockchain Technology, *Fourth International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT)*, Mandya, India, 2022, pp. 1–5. https://doi.org/10.1109/ICERECT56837.2022.10060287

[5] Z. Wang, L. Yang, Q. Wang, D. Liu, Z. Xu, S. Liu, ArtChain: Blockchain-Enabled Platform for Art Marketplace, *IEEE International Conference on Blockchain*, Atlanta, GA, USA, 2019, pp. 447–454. https://doi.org/10.1109/Blockchain.2019.00068

[6] H. Wang, D. He, X. Wang, C. Xu, W. Qiu, Y. Yao, Q. Wang, An Electricity Cross-Chain Platform Based on Sidechain Relay, *Journal of Physics: Conference Series*, Vol. 1631, No. 1, Article No. 012189, September, 2020. https://doi.org/10.1088/1742-6596/1631/1/012189

[7] C. Li, G. Zhang, X. Mao, J. Zhang, C. Xing, Multi-Chain Model and Cross-Chain Communication Protocol for Financial Transactions, *IEEE 22nd International Conference on Software Quality, Reliability, and Security Companion (QRS-C)*, Guangzhou, China, 2022, pp. 547–551. https://doi.org/10.1109/QRS-C57518.2022.00087

[8] L. Kan, Y. Wei, A. H. Muhammad, W. Siyuan, L. C. Gao, H. Kai, A Multiple Blockchains Architecture on Inter-Blockchain Communication, *IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Lisbon, Portugal, 2018, pp. 139–145. https://doi.org/10.1109/QRS-C.2018.00037

[9] Y. Pang, A New Consensus Protocol for Blockchain Interoperability Architecture, *IEEE Access*, Vol. 8, pp. 153719–153730, August, 2020. https://doi.org/10.1109/ACCESS.2020.3017549

[10] N. Shadab, F. Houshmand, M. Lesani, Cross-Chain Transactions, *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, Toronto, Canada, 2020, pp. 1–9. https://doi.org/10.1109/ICBC48266.2020.9169477

[11] L. Cao, B. Song, Blockchain Cross-Chain Protocol and Platform Research and Development, *International Conference on Electronics, Circuits and Information Engineering (ECIE)*, Zhengzhou, China 2021, pp. 264–269. https://doi.org/10.1109/ECIE52353.2021.00063

[12] Y. He, C. Zhang, B. Wu, Y. Yang, K. Xiao, H. Li, Cross-Chain Trusted Service Quality Computing Scheme for Multichain-Model-Based 5G Network Slicing SLA, *IEEE Internet of Things Journal*, Vol. 10, No. 14, pp. 12126–12139, July, 2023. https://doi.org/10.1109/JIOT.2021.3132388

[13] Z. Liu, Y. Xiang, J. Shi, P. Gao, H. Wang, X. Xiao, B. Wen, Q. Li, Y.-C. Hu, Make Web 3.0 Connected, *IEEE Transactions on Dependable and Secure Computing*, Vol. 19, No. 5, pp. 2965–2981, September-October, 2022. https://doi.org/10.1109/TDSC.2021.3079315

[14] J. Jiang, Y. Zhang, Y. Zhu, X. Dong, L. Wang, Y. Xiang, DCIV: Decentralized cross-chain data integrity verification with blockchain, *Journal of King Saud University – Computer and Information Sciences*, Vol. 34, No. 10, pp. 7988–7999, November, 2022. https://doi.org/10.1016/j.jksuci.2022.07.015

[15] J. Kwon, *Tendermint: Consensus Without Mining*, Technical Report, Tendermint, 2014. https://tendermint.com/static/docs/tendermint.pdf

[16] E. Buchman, J. Kwon, Z. Milosevic, The Latest Gossip on BFT Consensus, *arXiv preprint*, arXiv:1807.04938, July, 2018. https://arxiv.org/abs/1807.04938

[17] O. H. Karam, Pruning Generalized Hypercube Interconnection Networks for Diameter Preservation: RedCube, *International Conference on Computer and Applications (ICCA)*, Beirut, Lebanon, 2018, pp. 364–368. https://doi.org/10.1109/COMAPP.2018.8460415

[18] X. He, H. Liu, Node-to-Node Disjoint Paths in Hypercube Networks with Faulty Edges, *IET International Conference on Information and Communications Technologies (IETICT 2013)*, Beijing, China, 2013, pp. 335–339. https://doi.org/10.1049/cp.2013.0067

[19] M. Chatti, S. Yehia, C. Timsit, S. Zertal, A Hypercube-Based NoC Routing Algorithm for Efficient All-to-All Communications in Embedded Image and Signal Processing Applications, *International Conference on High Performance Computing & Simulation (HPCS)*, Caen, Normandy, France, 2010, pp. 623–630. https://doi.org/10.1109/HPCS.2010.5547065

[20] M. Taverna, K. G. Paterson, Snapping Snap Sync: Practical Attacks on Go Ethereum Synchronising Nodes, *32nd USENIX Security Symposium (USENIX Security'23)*, Anaheim, CA, USA, 2023, pp. 3331–3348.

[21] B. B. A. Christyono, M. Widjaja, A. Wicaksana, Go-Ethereum for electronic voting system using clique as proof-of-authority, *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, Vol. 19, No. 5, pp. 1565–1572, October, 2021. http://doi.org/10.12928/telkomnika.v19i5.20415

[22] S. Heron, Advanced Encryption Standard (AES), *Network Security*, Vol. 2009, No. 12, pp. 8–12, December, 2009. https://doi.org/10.1016/S1353-4858(10)70006-4

[23] M. Haekal, Eliyani, Token-Based Authentication Using JSON Web Token on SIKASIR RESTful Web Service, *International Conference on Informatics and Computing (ICIC)*, Mataram, Indonesia, 2016, pp. 175–179. https://doi.org/10.1109/IAC.2016.7905711

[24] M. Jones, J. Hildebrand, *Json Web Encryption (JWE)*, Report No. RFC 7516, May, 2015. https://datatracker.ietf.org/doc/html/rfc7516

[25] M. Jones, *JSON Web Key (JWK)*, Report No. RFC 7517, May, 2015. https://datatracker.ietf.org/doc/html/rfc7517

[26] Hypercube, Wikipedia, The Free Encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/Hypercube. [Accessed: 10 Aug 2023].

Biographies

Ekkarat Boonchieng received the B.S. (Computer Science) from Khon Kaen University, M.S. (Computer Science) from the University of New Haven, and Ph.D. (Computer Science) from the Illinois Institute of Technology. His research interests include computer network, image processing, blockchain, data science and biomedical engineering.

Chaimade Busayatananphon is a Ph.D. student in the Department of Computer Science at Chiang Mai University, Thailand. His research focuses on Blockchain technology and Artificial Intelligence, aiming to contribute to advancements in secure, intelligent systems through innovative academic inquiry and applied computational methodologies.

Sukhuntha Osiriphun is a food science researcher at Chiang Mai University. Her work focuses on food safety, blockchain applications in agro-industry, and innovative food processing technologies.

Anusorn Chaikaew is a Ph.D. candidate in the Department of Computer Science at Chiang Mai University, Thailand. His research interests include Blockchain, Ontology, Knowledge Graphs, and Artificial Intelligence, with a focus on integrating semantic technologies and decentralized systems to enhance intelligent data representation, reasoning, and interoperability.