CROSS DOMAIN AUTHENTICATION USING ORACLE

A PROJECT REPORT

Submitted by

SUBHASHINI V

(2023246028)

A report for the phase-I of the project
submitted to the Faculty of

INFORMATION AND COMMUNICATION ENGINEERING

in partial fulfillment
for the award of the degree
of

MASTER OF TECHNOLOGY
in
INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI 600 025

DECEMBER 2024

ANNA UNIVERSITY

CHENNAI - 600 025

BONA FIDE CERTIFICATE

Certified that this project report titled CROSS DOMAIN AUTHENTICATION USING ORACLE is the bona fide work of SUBHASHINI V who carried out project work under my supervision. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on this or any other candidate.

PLACE:	**DR. K. INDRA GANDHI**
DATE:	**ASSOCIATE PROFESSOR**
	PROJECT GUIDE
	DEPARTMENT OF IST, CEG
	ANNA UNIVERSITY
	CHENNAI 600025

COUNTERSIGNED

DR. S. SWAMYNATHAN
HEAD OF THE DEPARTMENT
DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI 600025

ABSTRACT

The current authentication systems rely heavily on Public Key Infrastructure (PKI) and consortium blockchains. PKI ensures secure communication within a single domain through trusted certificate authorities (CAs), but it fails in cross-domain authentication due to identity isolation. Consortium blockchains attempt to address this issue but face high operational costs, privacy concerns, and scalability limitations. These centralized models introduce single points of failure and require trust in third-party authorities, making them less sustainable for multi-domain environments.

To overcome these limitations, the proposed system employs a decentralized oracle-based cross-domain authentication scheme. It replaces the consortium blockchain model with blockchain oracles that facilitate secure and trustless data exchange between independent blockchains. Domain-specific smart contracts manage user identities, while zero-knowledge proofs enhance privacy. The system ensures tamper-proof and decentralized authentication by integrating oracles to verify credentials across domains without exposing sensitive information.

The implementation of this system uses oracles which results in a secure, scalable and privacy-preserving cross-domain authentication framework. It enables seamless communication between blockchains while protecting user data from potential breaches. Security analysis and experimental evaluations demonstrate its resilience against common cyberattacks, low operational costs and reduced authentication latency. This system advances blockchain interoperability, paving the way for a more integrated and trustworthy blockchain ecosystem.

ABSTRACT TAMIL

தற்போதைய அங்கீகார அமைப்புகள் பொதுக் குறி அத்தியமைப்பு (PKI) மற்றும் கூட்டமைப்பு பிளாக்செயின் (consortium blockchains) மீது அதிகமாக சார்ந்துள்ளது. PKI ஒரு தனி டொமைனில் நம்பகமான சான்றிதழ் ஆணையங்களின் (CA) மூலம் பாதுகாப்பான தொடர்பை உறுதி செய்கின்றது. ஆனால், அடையாள தனிமைப்படுத்தலால் இது இடம் பெற்றது, இது இடைவெளித் துறைகளுக்கான அங்கீகாரத்தில் தோல்வியடைந்துவிடுகிறது. கூட்டாட்சித் பிளாக்செயின் இந்த பிரச்சினையை தீர்க்க முயற்சி செய்கின்றன. ஆனால், அதிக செயல்பாட்டு செலவுகள், தனியுரிமை பிரச்சினைகள் மற்றும் பரவல்தன்மை குறைவால் அவை சிக்கல்களை சந்திக்கின்றன. இந்த மையப்படுத்தப்பட்ட மாதிரிகள் மூன்றாம் தரப்பு ஆணையங்களில் நம்பிக்கையை தேவைப்படுத்துகின்றன, மேலும் அவை ஒற்றை புள்ளி தோல்விகள் (single points of failure) அறிமுகப்படுத்துகின்றன, இதனால் பல துறைச் சுற்றுச்சூழலுக்கு அவை நிலைத்தன்மையற்றவையாக இருக்கின்றன.

இந்தக் குறைகளை மீறுவதற்காக, முன்மொழியப்பட்ட அமைப்பு மையமற்ற திருத்துக்களைக் கொண்டு இடைத் துறை அங்கீகார முறைமையை செயல்படுத்துகிறது. இது கூட்டாட்சித் பிளாக்செயின் மாதிரியை தளத்தட திருத்துக்களுடன் மாற்றுகிறது. இந்த திருத்துக்கள் தனித்தளத் தளத்தடங்களுக்கிடையே பாதுகாப்பான மற்றும் நம்பகமற்ற தரவு பரிமாற்றத்தை ஏற்படுத்துகின்றன. துறைக்கு உரிய ஸ்மார்ட் ஒப்பந்தங்கள் (smart contracts) பயனர் அடையாளங்களை நிர்வகிக்கின்றன, மேலும் பூஜ்ஜிய அறிவு சான்றுகள்(zero-knowledgeproofs) தனியுரிமையை மேம்படுத்துகின்றன. இந்த அமைப்பு துறைகளுக்கிடையே அங்கீகாரத்தை பாதுகாப்பாகவும் மையமற்றதாகவும் செயல்படுத்துகிறது, இதில் தரவுகளை அம்பலப்படுத்தாமல் திருத்துக்கள் சான்றுகளின் நம்பகத்தன்மையை உறுதி செய்கின்றன.

இந்த அமைப்பின் செயல்படுத்தல் ஆரக்கிள்ஸைப் பயன்படுத்துகிறது, இதன் விளைவாக ஒரு பாதுகாப்பான, அளவிடக்கூடிய மற்றும் தனியுரிமை-பாதுகாக்கும் இடைத் - துறை அங்கீகார கட்டமைப்பை உருவாக்குகிறது. இது பிளாக்செயின்களுக்கு இடையே எளிய தொடர்பை ஏற்படுத்துகிறது, மேலும் பயனர் தரவுகளை மீறல் சம்பவங்களிலிருந்து பாதுகாக்கின்றது. பாதுகாப்பு, பகுப்பாய்வு மற்றும் பரிசோதனை மதிப்பீடுகள் இதை பொதுவான கையகத்திற்குரிய தாக்குதல்களுக்கு எதிராகத் தடுப்பதாகவும், குறைந்த செயல்பாட்டு செலவுகளுடனும், குறைந்த அங்கீகார தாமதத்துடனும் இருப்பதை நிரூபிக்கின்றன. பிளாக்செயின் இயங்குநிலையை மேம்படுத்துகிறது, மேலும் ஒருங்கிணைக்கப்பட்ட மற்றும் நம்பகமான பிளாக்செயின் சுற்றுச்சூழல் அமைப்புக்கு வழி வகுக்கிறது.

ACKNOWLEDGEMENT

It is my privilege to express my deepest sense of gratitude and sincere thanks to Dr. K. INDRA GANDHI, Associate Professor, Project Guide, Department of Information Science and Technology, College of Engineering, Guindy, Anna University, for her constant supervision, encouragement, and support in my project work. I greatly appreciate the constructive advice and motivation that was given to help me advance my project in the right direction.

I am grateful to Dr. S. SWAMYNATHAN, Professor and Head, Department of Information Science and Technology, College of Engineering Guindy, Anna University for providing us with the opportunity and necessary resources to do this project.

I would also wish to express my deepest sense of gratitude to the Members of the Project Review Committee: Dr. S. SRIDHAR, Professor, Dr. G. GEETHA, Associate Professor, Dr. D. NARASHIMAN, Teaching Fellow Department of Information Science and Technology, College of Engineering Guindy, Anna University, for their guidance and useful suggestions that were beneficial in helping me improve my project.

I also thank the faculty member and non-teaching staff members of the Department of Information Science and Technology, Anna University, Chennai for their valuable support throughout the course of our project work.

SUBHASHINI V
(2023246028)

TABLE OF CONTENTS

	ABSTRACT	iii
	ABSTRACT TAMIL	iv
	LIST OF TABLES	viii
	LIST OF FIGURES	ix
	LIST OF SYMBOLS AND ABBREVIATIONS	x
1	**INTRODUCTION**	**1**
	1.1 BLOCKCHAIN	1
	1.2 ORACLE	2
	1.3 SMART CONTRACTS	3
	1.4 CROSS DOMAIN AUTHENTICATION	4
	1.5 OBJECTIVE	5
	1.6 CHALLENGES IN EXISTING SYSTEMS	5
	1.7 PROPOSED SYSTEM	6
	1.8 ORGANIZATION OF THE REPORT	7
2	**LITERATURE SURVEY**	**8**
	2.1 RELATED WORK	8
	2.1.1 Blockchain based authentication	8
	2.1.2 Blockchain Oracle	10
	2.1.3 Hybrid blockchain	11
	2.1.4 Cross Domain Authentication	12
	2.2 SUMMARY OF THE ISSUES	14
3	**SYSTEM ARCHITECTURE OF CROSS-DOMAIN AUTHENTICATION USING ORACLE**	**16**
	3.1 SYSTEM DESIGN	16
	3.1.1 Initialization Phase	17
	3.1.2 Authentication Phase	18
	3.1.3 Key Agreement Phase	19
	3.1.4 Revocation Phase	20
	3.2 SOFTWARE REQUIREMENT	21
	3.2.1 Truffle	21
	3.2.2 Ganache	21

	3.2.3 Web3.js	21
	3.2.4 MetaMask	22
	3.2.5 Chainlink	23
	3.2.6 Docker	23
	3.3 HARDWARE REQUIREMENT	24
4	**IMPLEMENTATION**	**25**
	4.1 BLOCKCHAIN DEVELOPMENT AND DEPLOYMENT	25
	4.1.1 Vehicle Details Blockchain	25
	4.1.2 Vehicle Insurance Blockchain	28
	4.2 ORACLE IMPLEMENTATION	31
	4.2.1 Chainlink node installation	31
	4.2.2 Oracle smart contract deployment	31
5	**RESULTS AND ANALYSIS**	**33**
	5.1 VEHICLE DETAILS USER REGISTRATION PAGE	33
	5.2 METAMASK TRANSACTION REQUEST FOR VEHICLE DETAILS REGISTRATION	33
	5.3 TRANSACTION DETAILS IN GANACHE FOR VEHICLE DETAILS BLOCKCHAIN	34
	5.4 VEHICLE INSURANCE USER REGISTRATION PAGE	34
	5.5 METAMASK TRANSACTION REQUEST FOR VEHICLE INSURANCE REGISTRATION	35
	5.6 TRANSACTION DETAILS IN GANACHE FOR VEHICLE INSURANCE BLOCKCHAIN	36
	5.7 FETCHING DATA AND DISPLAYED USING ORACLE	37
	5.8 PERFORMANCE ANALYSIS	37
	5.9 CHALLENGES IN IMPLEMENTATION	39
6	**CONCLUSION AND FUTURE WORK**	**40**
	6.1 CONCLUSION	40
	6.2 FUTURE WORK	41
REFERENCES		**42**

LIST OF TABLES

5.1 Comparisons of Functionalities of Different Systems 38

LIST OF FIGURES

1.1 Generic model of oracle feeding data into blockchain 2

3.1 Architecture Diagram 16

4.1 Generated key for Vehicle Details Blockchain 26
4.2 Contract compilation 26
4.3 Migration 27
4.4 Ganache configuration 29
4.5 Generated key for Vehicle Insurance Blockchain 29
4.6 Compiling Vehicle Insurance smart contract 30
4.7 Smart contract deployment 30

5.1 Vehicle Details user registration page 33
5.2 Metamask transaction request for Vehicle Details 34
5.3 Transaction details in ganache for vehicle details 34
5.4 Vehicle Insurance user registration page 35
5.5 Metamask transaction request for Vehicle Insurance 36
5.6 Transaction details in ganache for vehicle insurance 36
5.7 Data fetched and displayed using oracle 37
5.8 Time cost for one user 38

LIST OF SYMBOLS AND ABBREVIATIONS

PKI	Public Key Infrastructure
CA	Certificate Authorities
DOS	Decentralized Oracle System
ABI	Application Binary Interface

CHAPTER 1

INTRODUCTION

1.1 BLOCKCHAIN

Blockchain is a decentralized, distributed ledger technology that records transactions across a network of computers in a secure, transparent, and tamper-proof manner, ensuring all the nodes share responsibility for data integrity and transparency. It operates without a central authority, distributing control among participants, and organizes data into blocks, each containing a group of transactions along with a cryptographic hash of the previous block, creating a secure and immutable chain. This structure ensures immutability, as altering a single block would invalidate the entire chain, making it extremely resilient against tampering or unauthorized modifications. Blockchain maintains trust and accountability through transparency, enabling all participants to view transactions and verify the data independently. The decentralized nature of blockchain eliminates single points of failure and reduces the risk of fraud or malicious interference, as data is redundantly stored across multiple nodes in the network. This redundancy enhances both security and reliability, making it particularly well-suited for applications where trust and integrity are paramount.

To validate transactions and add new blocks to the chain, networks employ consensus mechanisms such as Proof of Work (PoW), which requires computational power to solve complex mathematical puzzles, or Proof of Stake (PoS), which leverages participants' stake in the network to ensure honest behavior. These mechanisms ensure agreement across distributed nodes and prevent double-spending or malicious activity. Blockchain technology has wide-ranging applications beyond cryptocurrency, including supply chain management, healthcare, identity verification, and voting systems. Moreover, blockchain supports interoperability through advanced frameworks and protocols, allowing multiple blockchains to communicate and exchange information seamlessly. This interoperability fosters collaboration across industries and opens new avenues for innovation. As the technology evolves, developments in scalability, energy efficiency, and security continue to drive its adoption in both private and public sectors.

1.2 ORACLE

A blockchain oracle is a mechanism that acts as a bridge between blockchain systems and external data sources, enabling smart contracts to access and interact with off-chain information. Blockchains are inherently closed systems, designed to ensure security and immutability, but this isolation limits their ability to utilize real-world data. Oracles overcome this limitation by securely fetching, verifying, and delivering data to smart contracts as shown in the Figure 1.1. Particularly, a decentralized oracle further enhances reliability by aggregating data from multiple independent sources, which reduces the risk of inaccuracies or manipulation. By distributing trust across a network of oracles, these systems minimize vulnerabilities and eliminate reliance on a single point of failure, enabling smart contracts to execute based on reliable information.

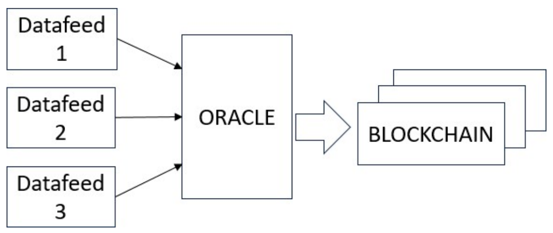


Figure 1.1: Generic model of oracle feeding data into blockchain

This architecture ensures that even in the presence of malicious actors or faulty data sources, the overall system remains trustworthy and accurate. Decentralized oracles support a wide range of applications, including financial data feeds for DeFi protocols, weather information for insurance contracts, and supply chain tracking for verifying product origins. Furthermore, the use of cryptographic techniques and consensus mechanisms by oracles ensures that data integrity and authenticity are preserved throughout the process. As blockchain technology continues to evolve, oracles play a critical role in expanding the functionality of decentralized ecosystems. They bridge the gap between isolated blockchains and the dynamic real world, unlocking new opportunities for innovation while preserving the core principles of security and decentralization.

1.3 SMART CONTRACTS

Smart contracts are self-executing digital agreements with terms written directly into blockchain networks, the specific conditions automatically trigger and execute enforcement once achieved. They eliminate the need for an intermediary, ensuring transparency, security and efficiency. Written in programming languages such as Solidity or Rust, they are deployed on blockchains to be immutable and tamper-proof. Once deployed, these contracts operate autonomously, requiring no further human intervention, and their results are securely recorded on the blockchain ledger. Their immutability ensures that once created, their code and logic cannot be altered, preventing unauthorized modifications and fostering trust among participants. Additionally, the use of blockchain cryptography guarantees that transactions and operations remain secure and verifiable by all parties. Some of the key features of smart contracts include autonomy, transparency and trustless operation, as they can function independently once deployed. Their transparent nature allows all stakeholders to verify the contract’s logic and execution, removing ambiguities and promoting fairness. They also significantly reduce costs by eliminating the need for intermediaries such as banks, lawyers or brokers. Smart contracts are revolutionizing traditional systems by streamlining processes, reducing operational inefficiencies and enhancing trust through decentralized execution. As blockchain technology continues to mature, smart contracts are set to play an integral role in shaping the future of digital agreements and decentralized ecosystems.

1.4 CROSS DOMAIN AUTHENTICATION

In a blockchain ecosystem, cross-domain authentication refers to the process of securely verifying and validating identities, credentials, or transactions across multiple independent blockchain networks or domains. As blockchain applications grow in complexity, interoperability between separate networks (domains) has become a critical challenge. Each blockchain may operate with its own consensus protocols, governance models, and authentication mechanisms, creating challenges that limit seamless communication and trust between systems. Cross-domain authentication addresses these challenges by enabling trustless interaction between networks while preserving the security, decentralization and integrity of each domain. This allows a user or entity authenticated on one blockchain to be securely recognized and trusted by another blockchain. The process eliminates the need for centralized intermediaries, ensuring that the interaction remains transparent, decentralized and tamper-proof. To achieve this, cross-domain authentication relies on innovative techniques such as decentralized oracles, cryptographic proofs and interoperability protocols.

These mechanisms facilitate secure data exchange and validation across networks without exposing sensitive information. Cross-domain authentication not only enhances user experience by reducing redundancies and delays but also opens up opportunities for collaborative applications across industries. It paves the way for a more unified and scalable blockchain ecosystem, supporting seamless interaction and fostering innovation across diverse blockchain platforms. This approach is crucial for the future of blockchain as it transitions from isolated systems to a fully interoperable, decentralized network of networks.

1.5 OBJECTIVE

The objective of this project is to develop a robust, scalable, and privacy-preserving framework for cross-domain communication between blockchain networks. The project focuses on facilitating seamless data exchange between the Vehicle Detail Blockchain and the Vehicle Insurance Blockchain, enabling critical operations such as insurance verification during vehicle registration. It aims to enhance privacy and security by incorporating mechanisms like zero-knowledge proofs and pseudo-identities to protect sensitive user data. A decentralized oracle network will be designed and integrated to act as an intermediary for fetching, validating and transmitting data across the blockchains, eliminating reliance on centralized entities. Furthermore, the project focus on improving efficiency and scalability by utilizing existing blockchain infrastructures, minimizing operational costs and ensuring support for high transaction volumes. Smart contracts will be deployed on both blockchains to automate processes like data validation and updates which provide efficiency and reliability. This project is intended to bridge blockchain domains securely using oracle while optimizing privacy, scalability and operational effectiveness.

1.6 CHALLENGES IN EXISTING SYSTEMS

Existing systems for cross-domain authentication face significant challenges that limit their efficiency and security. Public Key Infrastructure (PKI) systems effectively manage intra-domain authentication but fail to facilitate cross-domain identity recognition due to the isolation of identity management across domains. Many solutions depend on consortium blockchains, which not only require re-registration of users but also entail high implementation costs and complex coordination efforts. Privacy concerns further exacerbate the issue, as sensitive data shared across domains in consortium blockchains is vulnerable to breaches, with a single misconfigured node capable of exposing critical information. Scalability also becomes a challenge as the number of domains increases, creating inefficiencies in maintaining consensus and reducing overall system performance. Moreover, the reliance on third-party Certificate Authorities (CAs) introduces security vulnerabilities such as collusion, unfair practices and single points of failure. Additionally, existing systems often lack robust privacy-preservation mechanisms, leaving sensitive user data exposed during authentication processes and increasing the risk of identity theft and malicious attacks.

1.7 PROPOSED SYSTEM

This system introduces a privacy-preserving cross-domain authentication scheme that leverages blockchain technology, smart contracts, and decentralized oracle systems (DOS) to enable seamless and secure communication between independent blockchain domains. The system addresses key challenges such as privacy protection, scalability, and interoperability while eliminating the reliance on centralized intermediaries. Each domain operates independently with its own Key Generation Center for system initialization and user registration, where credentials are securely stored on the blockchain using smart contracts. During the authentication process, users from one domain initiate authentication requests to another domain through the DOS, which acts as a trusted intermediary by securely relaying, verifying, and aggregating data to prevent manipulation. To preserve user privacy, zero-knowledge proof was employed, allowing users to authenticate without revealing sensitive information. Additionally, the system includes a revocation phase to erase or revoke user credentials when necessary, ensuring security in dynamic environments. The decentralized oracle system plays a critical role in bridging trust between domains, enabling secure data exchange while mitigating risks like tampering and single points of failure. Overall, the proposed system enhances security, scalability, and cost efficiency by integrating decentralized technologies, ensuring privacy and fostering trustless interoperability across multiple blockchain domains.

1.8 ORGANIZATION OF THE REPORT

This report is organized into 6 chapters, describing each part of the project with detailed illustrations and system design diagrams.

Chapter 2: Related works reviews existing methods, technologies and research gaps related to blockchain, cross-authentication and decentralized oracle systems.

Chapter 3: System Architecture and Design describes the architecture, phases and components of the proposed system.

Chapter 4: Implementation explains the tools, technologies and processes used to implement the proposed system.

Chapter 5: Results of the proposed system, analyzes results and compares it with existing solutions.

Chapter 6: Conclusion and Future Work summarizes key findings, highlights contributions and suggests future research directions.

CHAPTER 2

LITERATURE SURVEY

2.1 RELATED WORK

Decentralized authentication schemes have gained significant attention in recent years with blockchain technology being a key enabler for achieving cross-domain authentication and addressing the challenges of centralization. Blockchain’s decentralized and distributed nature makes it a suitable foundation for secure and transparent cross-domain interactions. Numerous blockchain-based cross-domain authentication mechanisms have been proposed, each contributing unique approaches to solving the issues of security, privacy and efficiency.

2.1.1 Blockchain based authentication

Shen et al. [1] introduced an efficient and secure blockchain-based authentication mechanism specifically designed for Internet of Things (IoT) devices operating across different domains. To address the inherent storage limitations of blockchain systems, the authors proposed an off-chain storage method. This solution, while reducing the on-chain data burden requires the explicit identity of entities during the verification process. The downside of this requirement is that it risks exposing sensitive identity information. Moreover, the scheme primarily focuses on the security framework without providing formal logical and security proofs which weakens the overall trust in the robustness of the proposed solution.

In an effort to enhance blockchain throughput and protect user privacy, the XAuth mechanism was proposed in [2]. XAuth integrates multiple Merkle hash trees with the InterPlanetary File System (IPFS) to improve the overall scalability of the blockchain system. Additionally, XAuth employs Pinocchio-based zero-knowledge proof (ZKP) techniques to achieve anonymous authentication, ensuring that user identities remain confidential. While this method addresses privacy concerns effectively, the introduction of zero-knowledge proofs imposes a heavy computational overhead on the system. This high computational demand could be a bottleneck in environments where low-latency processing is critical.

Akhilesh et al. [3] proposed an innovative blockchain-based protocol called Proof-by-Approval (PbA) to address inefficiencies in secure banking transactions. The protocol enhances security and decentralization while reducing the resource-intensive nature of traditional consensus methods like Proof-of-Work (PoW). PbA introduces a dual-approval mechanism where users create transaction blocks that must be validated and approved by authorized banking entities, termed ”Approvers,” before being added to the blockchain. The system leverages cryptographic tools, such as RSA and SHA256 hashing, for robust data integrity and verification. Hyperledger Fabric is employed to manage a private, permissioned blockchain, ensuring flexibility and modularity for real-world banking applications.While the protocol reduces the computational overhead and energy consumption compared to PoW, it has notable limitations. The dual-approval process can introduce latency, and reliance on centralized approvers slightly undermines the decentralization ideal. Moreover, setting up a PbA-based blockchain with strict validation mechanisms can be complex and resource-intensive for large-scale banking systems

Srivastava et al. [4] proposed a secure decentralized identity management model leveraging blockchain technology, self-sovereign identity principles and oracle technology. This approach addresses challenges such as cross-domain authentication, lack of credibility in identity verification and security vulnerabilities in centralized systems. The system employs Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) for user authentication, integrating identity blockchains like Hyperledger Indy with transaction ledgers such as Hyperledger Fabric. Key features include enhanced security using a modified Elliptic Curve Digital Signature Algorithm (ECDSA), decentralized key management, and robust threshold cryptography. While the model enhances security, privacy, and interoperability, it involves complex setups and may face scalability challenges as system participants grow.

Yuan and Wang [5] introduced a six-layer reference model for blockchain systems, which aims to standardize blockchain architecture and highlight its key components, including data, network, consensus, incentive, contract and application layers. This model provides a structured framework to analyze blockchain systems, emphasizing features such as decentralization, data integrity and programmability. The framework offers advantages like improved traceability, enhanced system organization and applicability across diverse domains. However, the model has limitations, such as the complexity of implementing the six-layer design in real-world scenarios and the absence of a detailed evaluation for scalability and efficiency in large-scale blockchain networks.

2.1.2 Blockchain Oracle

Alhussayen et al. [6] proposed a blockchain oracle-based interoperability technique for permissioned blockchains, designed to enable seamless cross-network transactions, including both read and write operations, between platforms like Hyperledger Fabric and Corda. The method leverages oracle service nodes to handle communication, transaction formatting, validation, and rollback mechanisms for maintaining ledger consistency.

This approach eliminates the need for modifications to the participating blockchain platforms and preserves decentralization. However, while the method demonstrates acceptable transaction latency, interactions with slower platforms like Corda result in higher delays. Additionally, the evaluation lacks scalability testing with larger network setups, and the system relies on trust between oracle nodes and the blockchain networks, which may pose challenges in highly adversarial environments.

Tong et al. [7] proposed an interoperability solution for blockchain-based Internet of Vehicles (BIoV), leveraging a multiple-oracle network to facilitate data transportation between homogeneous/heterogeneous blockchains and off-chain data sources. The system uses consensus algorithms like RAFT and PBFT for voting mechanisms, ensuring the authenticity of data amidst potential malicious attacks. This approach supports decentralized data validation and repacking before feeding verified information back to target blockchains. While the model offers high security and scalability, it has limitations, including increased computational overhead for complex interoperations and challenges in maintaining performance when scaling to larger networks with diverse data sources.

2.1.3 Hybrid blockchain

An approach presented in [8] uses a hybrid blockchain model to achieve cross-domain authentication. This scheme implements a multi-signature smart contract on a public consortium blockchain which facilitates authentication through both symmetric and asymmetric encryption methods. These encryption techniques are employed to safeguard message privacy and enable key agreement. However, the dual-encryption approach significantly increases communication overhead which poses a challenge for resource-constrained devices, such as drones. The heavy communication requirements could limit the practical applicability of this scheme in environments where low-power devices are prevalent.

Z. Cui et al. [9] introduced multi-tier network structure, dividing IoT nodes into base stations, cluster head nodes and ordinary nodes. The scheme uses a hybrid blockchain model with local blockchains for cluster head nodes and a public blockchain for base stations. Identity authentication is managed in a decentralized manner, with ordinary nodes authenticated locally and cross-domain communications facilitated through the public blockchain.The limitations include the overhead and latency introduced by maintaining both local and public blockchains, especially under high network traffic. Additionally, energy consumption for message transmission and cryptographic operations may challenge the efficiency of nodes with limited resources.

2.1.4 Cross Domain Authentication

In [10], The multi-CA-based cross-domain authentication scheme was introduced that uses blockchain technology to establish decentralized trust among multiple domains. Each domain operates its own Certificate Authority (CA) for certificate management, including issuance, updating and revocation utilizing a consortium blockchain for secure interactions. A unique revocation mechanism based on the Poisson process balances efficiency and storage needs. Privacy is ensured through cryptographic techniques such as hashing and signature verification. However, the system faces scalability challenges due to the complexity of managing certificates across domains and the computational costs of cryptographic operations. Additionally, delays in updating and revoking certificates within the distributed ledger can impact overall efficiency.

In [11], a cross-domain authentication mechanism tailored for the healthcare sector was introduced. The scheme focuses on the interaction between hospitals and patients across medical consortium blockchains. To enable user revocation in this healthcare context, an improved chameleon hash function is utilized. While the scheme demonstrates an effective solution for user revocation and cross-chain interaction, it relies on the presence of a trusted hospital acting as a third party. This requirement for a trusted intermediary weakens the decentralized nature of the system and introduces potential risks if the third-party entity is compromised.

In [12], the proposed methodology involves constructing an efficient Cross-Domain Handshake Scheme (CDHS) for Mobile Healthcare Social Networks (MHSNs). The methodology uses elliptic curve cryptography (ECC) and is designed to enable two users from different healthcare centers to establish secure communications through a handshake with symptoms-matching. The system involves a trusted authority (TA) generating private keys for healthcare centers, which in turn issue private keys to patients. The scheme incorporates mutual authentication, patient anonymity, session key agreement, and resistance to common cyber-attacks, ensuring secure cross-domain communication. However, the limitations include computational complexity due to the use of bilinear pairings, challenges in practical deployment in real-world scenarios and potential performance issues on devices with lower computational capabilities.

Chai et al. [13] introduced the Blockchain-Based Secure Cross-Domain Data Access (BSCDA) scheme to address the complexities of secure data sharing in the Internet of Things (IoT). By leveraging blockchain's decentralization, transparency and tamper-proof capabilities, the scheme eliminates risks associated with single points of failure and builds trust among different domains. A key innovation of BSCDA is its certificate management method, which uses a hash index table to reduce the storage overhead typically associated with certificate revocation lists while maintaining verifiability. Additionally, the scheme incorporates a four-party key agreement mechanism, enabling the negotiation of secure session keys for safe data transmission across domains. To handle large-scale data storage efficiently, the scheme utilizes IPFS, storing encrypted data off-chain while recording hash addresses on the blockchain. This approach not only minimizes the blockchain's storage load but also ensures data security during transmission. Extensive security analysis demonstrated BSCDA's resistance to man-in-the-middle and eavesdropping attacks, while experimental results highlighted its scalability, low computational overhead, and suitability for real-world cross-domain IoT scenarios. However, a key limitation of the scheme lies in its reliance on IPFS for off-chain data storage, which introduces risks related to data availability and access delays, especially when the network experiences high traffic or disruptions.

2.2 SUMMARY OF THE ISSUES

Existing cross-domain authentication mechanisms face several critical issues that limit their effectiveness and applicability. Many solutions require explicit identity disclosure during verification, exposing sensitive user information and compromising privacy. Additionally, methods involving dual encryption or complex multi-tier architectures impose high communication overhead, making them impractical for resource-constrained devices such as IoT nodes or drones. The use of advanced techniques like zero-knowledge proofs and elliptic curve cryptography often leads to significant computational complexity, which limits their deployment in latency-sensitive or low-power environments. Furthermore, reliance on trusted third parties, such as hospitals or central authorities, undermines the decentralized nature of blockchain systems and introduces vulnerabilities if these intermediaries are compromised. Scalability remains a major challenge, as managing certificates or maintaining dual blockchain structures becomes increasingly complex with the addition of more users or domains. High network traffic and the need to support both local and public blockchains result in latency and energy inefficiency, particularly for devices with limited computational resources. Finally, mechanisms dependent on distributed ledgers often experience delays in updating and revoking certificates, reducing system responsiveness and overall efficiency. These challenges underscore the necessity for a balanced framework that ensures privacy, scalability, efficiency, and decentralization.

CHAPTER 3

SYSTEM ARCHITECTURE OF CROSS-DOMAIN AUTHENTICATION USING ORACLE

3.1 SYSTEM DESIGN

The architecture diagram for cross-domain authentication scheme between two independent blockchain domains that is the Vehicle Details Domain and the Vehicle Insurance Domain as shown in the Fig 3.1.

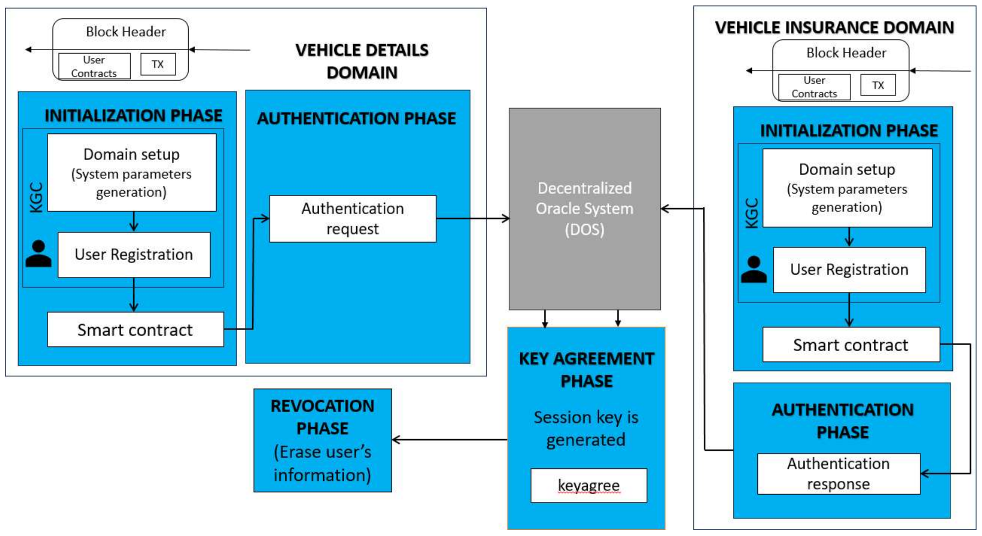


Figure 3.1: Architecture Diagram

It highlights the steps for secure communication using a Decentralized Oracle System (DOS). The architecture is divided into four distinct phases: Initialization Phase, Authentication Phase, Key Agreement Phase, and Revocation Phase. The Vehicle Details Domain is a blockchain-based system designed to store and manage all vehicle-related information with a focus on data accuracy, transparency, and security. It maintains critical details such as the registration number and owner information, and manufacturer specifications like model, make and colour of the vehicle. By leveraging blockchain features like immutability and decentralized storage, this system ensures trustworthy and easily verifiable vehicle data. The Vehicle Insurance Domain is a blockchain-based system dedicated to managing vehicle insurance policies, enhancing policy administration, claims processing and fraud prevention. It securely stores essential data such as insurance policy number, owner name, registration number, insurance amount and claimed status. By utilizing blockchain features like transparency, smart contracts and data integrity, this system ensures efficient, tamper-proof insurance management.

3.1.1 Initialization Phase

The Initialization Phase serves as the foundational step in establishing a secure cross-domain authentication scheme, ensuring that all users are properly registered and equipped with the necessary credentials before engaging in inter-domain communication. This phase includes three key operations: domain setup, user registration and smart contract deployment.

- **Domain setup**

  Each domain initiates the authentication process by having its Key Generation Center (KGC) generate essential system parameters. These parameters include cryptographic keys and security settings required for secure encryption, decryption and digital signatures. The KGC ensures that these cryptographic elements meet security standards necessary for protecting sensitive data during cross-domain interactions.

- **User registration**

Users must register with their respective domains to participate in the authentication scheme. During this process, the KGC issues cryptographic credentials, such as public/private key pairs, digital certificates or pseudo-identities. These credentials enable secure identity verification and encrypted communication. After successful registration, users' credentials and relevant information are securely stored in the blockchain through a registration transaction executed by a smart contract.

- **Smart contract deployment**

  Smart contracts play a crucial role by automating the registration process and enforcing predefined security policies. They ensure that only eligible users are registered while maintaining the integrity, transparency, and immutability of registration records. The smart contracts also act as tamper-proof repositories, preventing unauthorized modifications to user credentials. By completing these operations, the Initialization Phase guarantees that all users within the domains are securely registered, authenticated, and provisioned with appropriate credentials. This setup lays a strong foundation for secure and trustworthy cross-domain authentication.

3.1.2 Authentication Phase

The Authentication Phase establishes mutual trust between the two independent domains that is the Vehicle Details Domain and the Vehicle Insurance Domain which enables secure cross-domain interactions. This process begins when a user from the Vehicle Details Domain initiates an authentication request to access services or data from the Vehicle Insurance Domain. The request is transmitted securely through an encrypted communication channel to a Decentralized Oracle System (DOS), which serves as a trusted intermediary. The DOS plays a critical role as a decentralized trust bridge, ensuring that all data exchanged between the domains is verified, tamper-proof, and trustworthy. Upon receiving the authentication request, the DOS verifies its authenticity using blockchain-based validation protocols. After ensuring the request is legitimate, the DOS forwards it securely to the Vehicle Insurance Domain for further verification. The Vehicle Insurance Domain then processes the request by validating the user's credentials and permissions according to its security policies and blockchain records. Once the verification process is completed, the Vehicle Insurance Domain sends an authentication response back to the DOS, indicating whether the user is successfully authenticated or denied access. The DOS relays this authentication response back to the Vehicle Details Domain, completing the trust establishment process. By utilizing decentralized oracles, this phase eliminates the need for centralized intermediaries, ensuring a secure, transparent, and decentralized authentication mechanism between the two blockchain systems.

3.1.3 Key Agreement Phase

The Key Agreement Phase establishes a secure communication channel between users in the Vehicle Details Domain and the Vehicle Insurance Domain by generating a shared session key. This phase begins immediately after successful authentication. Both domains collaboratively generate a session key, ensuring that all subsequent communications are encrypted, confidential, and protected from tampering or interception. The Decentralized Oracle System (DOS) plays a pivotal role in this phase by facilitating the secure exchange of the session key between the two domains. As a trust anchor, the DOS ensures that the key agreement process is transparent, verifiable, and resistant to attacks such as key interception or manipulation. The decentralized nature of the DOS prevents any single point of failure, making the entire process highly secure. Once the session key is established, it is used to encrypt sensitive data exchanged between the domains, such as vehicle details, insurance policy information, and claim records. This encryption safeguards the data from unauthorized access and ensures its integrity throughout the communication process. By completing the Key Agreement Phase, both domains can securely exchange critical data, confident that their communications remain private, authenticated, and protected against potential threats. This phase guarantees the establishment of a secure channel for efficient and confidential cross-domain interactions.

3.1.4 Revocation Phase

The Revocation Phase addresses the secure removal of a user's access when they are no longer authorized to participate in the cross-domain authentication system. This process begins with securely revoking or erasing the user's credentials, including cryptographic keys, digital certificates, and registration data, ensuring the user can no longer access or interact with either the Vehicle Details Domain or the Vehicle Insurance Domain. This action prevents unauthorized users from exploiting the system after their permissions have been revoked. To maintain transparency and trust, the revocation process is recorded on the blockchain through a secure transaction. This blockchain update ensures that the user's revoked status is immutable, verifiable, and visible to relevant entities in the ecosystem. The decentralized nature of the blockchain prevents any tampering or rollback of the revocation records, reinforcing the integrity of the system. By implementing the Revocation Phase, the authentication framework ensures continuous security and trustworthiness by dynamically managing user access rights.

3.2 SOFTWARE REQUIREMENT

3.2.1 Truffle

Truffle is a widely-used development framework for Ethereum blockchain projects, designed to streamline the process of writing, testing, and deploying smart contracts. It automates tasks like compiling Solidity code and deploying contracts to the blockchain. Truffle offers built-in testing capabilities, allowing developers to write and run unit tests using JavaScript or Solidity. Its migration system ensures smart contracts are deployed in the correct order, while its interactive console provides a powerful environment for interacting with deployed contracts. Truffle also simplifies network management, making it easy to connect to local blockchains, testnets like Rinkeby, or the Ethereum mainnet. It is a critical tool for blockchain developers, especially when combined with Ganache for local testing.

3.2.2 Ganache

Ganache is a personal Ethereum blockchain that runs locally, making it an essential tool for testing and debugging smart contracts in a controlled environment. It provides pre-funded accounts, enabling developers to simulate transactions without spending real Ether. Ganache is highly configurable, allowing users to adjust parameters like gas limits or block times for testing edge cases. It features both a graphical user interface (Ganache UI) and a command-line tool (Ganache CLI) to suit different development preferences. Its detailed transaction logging and instant block mining capabilities make it invaluable for developers who need to debug and iterate on their smart contract code.

3.2.3 Web3.js

Web3.js is a JavaScript library that enables developers to interact with the Ethereum blockchain from web applications. It provides APIs for reading blockchain data, sending transactions, and deploying smart contracts. Through ABI files, Web3.js allows seamless integration with smart contracts, enabling function calls and event listening. It supports multiple connection protocols, including HTTP and WebSocket, and works seamlessly with wallets like MetaMask for transaction signing. Web3.js is a critical tool for building decentralized applications (dApps), as it bridges the frontend interface with the underlying blockchain, empowering developers to create dynamic, blockchain-enabled user experiences.

3.2.4 MetaMask

MetaMask is a browser extension and mobile app that acts as both a cryptocurrency wallet and a gateway to blockchain networks. It allows users to store, send, and receive Ether and ERC-20 tokens securely. MetaMask simplifies blockchain interaction by enabling seamless network switching between Ethereum mainnet, testnets, and custom networks like Ganache. It also facilitates transaction signing, allowing users to approve and send blockchain transactions directly from the wallet. MetaMask integrates with decentralized applications (dApps) through Web3.js, making it a crucial tool for interacting with blockchain-based systems during development and deployment. Its robust security measures, including private key encryption, make it a trusted choice for developers.

3.2.5 Chainlink

Chainlink is a decentralized oracle network that bridges the gap between blockchain smart contracts and off-chain data or services. It provides a secure, reliable way to fetch external data, such as API responses or real-world events, and deliver it on-chain. Developers can define job specifications for Chainlink nodes that perform tasks such as data retrieval or computation. Chainlink ensures data integrity by aggregating results from multiple sources, reducing the risk of tampering or errors. With pre-built smart contracts like ChainlinkClient.sol, integration is straightforward. Chainlink is commonly used for applications requiring price feeds, random number generation, or cross-chain communication, making it a cornerstone for decentralized systems.

3.2.6 Docker

Docker is a platform that enables the creation, deployment, and management of lightweight, portable containers that encapsulate applications and their dependencies, ensuring consistency across various environments. It simplifies development, testing, and deployment by providing isolation, efficiency, and scalability. In the context of running Chainlink nodes, Docker is highly valuable as it allows you to easily set up, deploy, and manage the necessary components like the Chainlink node, database (e.g., PostgreSQL), and blockchain client (e.g., Ethereum node) in isolated containers. This ensures compatibility and simplifies configuration, scaling, and updates, making Docker an essential tool for managing Chainlink infrastructure.

3.3 HARDWARE REQUIREMENT

- **Processor (CPU):** Intel Core i5 or AMD Ryzen 7.

- **Memory (RAM):** 16 GB or more(sufficient to handle multiple tools simultaneously, such as Truffle, MetaMask, and Chainlink nodes).

- **Storage:** 512 GB SSD or more (ensures efficient storage and retrieval of blockchain data).

CHAPTER 4

IMPLEMENTATION

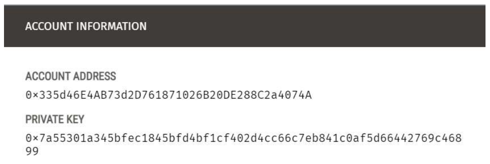
The implementation of the Cross-Domain Authentication using Oracle incorporates advanced blockchain technologies, decentralized oracles, and Web3 frameworks to achieve a robust, privacy-preserving and scalable solution. This section elaborates on the core stages of implementation, encompassing smart contract development, oracle configuration, deployment wallet integration, and cross-domain communication.

4.1 BLOCKCHAIN DEVELOPMENT AND DEPLOYMENT

4.1.1 Vehicle Details Blockchain

- **Truffle Initialization:** Truffle initialization refers to setting up a Truffle project for blockchain development. It involves running the command `truffle init` to generate a basic project structure, which includes the following directories and files:
    1. contracts/: Stores Solidity smart contracts (VehicleDetails.sol).
    2. migrations/: Contains deployment scripts for deploying contracts to a blockchain.
    3. truffle-config.js: Configuration file for setting up networks, compilers, and other Truffle options.

- **Key Generation:** In Ganache, key generation is the process of creating Ethereum accounts with private keys and associated public addresses. Ganache is a personal blockchain for Ethereum development that generates multiple accounts by default, each with its own private key and public address. The account address generated was 0x335d46e4ab73d2D761871826B20DE288C2a4074A and the private key was 0x7a55301a345bfec1845bfd4bf1cf402d4cc66c7eb84 1c0af5d66442769c46899



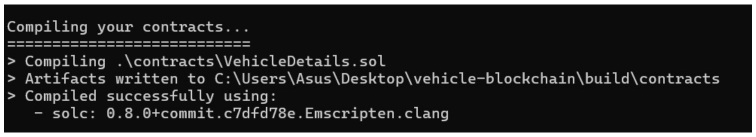
ACCOUNT INFORMATION

ACCOUNT ADDRESS
0×335d46E4AB73d2D761871026B20DE288C2a4074A

PRIVATE KEY
0×7a55301a345bfec1845bfd4bf1cf402d4cc66c7eb841c0af5d66442769c468
99

Figure 4.1: Generated key for Vehicle Details Blockchain

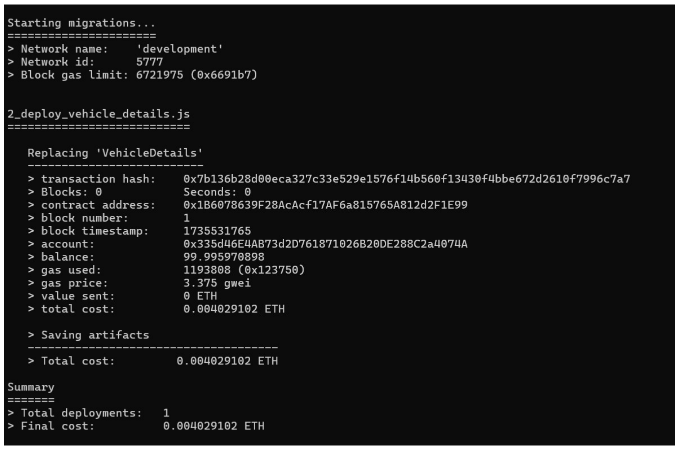
- **Truffle Compile:** The command truffle compile is used to compile Solidity smart contracts within a Truffle project. When executed, it scans the contracts/ directory for Solidity files (e.g., .sol files) and compiles them into .json artifacts stored in the build/contracts/ directory. These artifacts contain the contract's ABI (Application Binary Interface) and bytecode, which are crucial for deploying and interacting with the contracts. If no changes are detected in the contracts, Truffle optimizes the process by skipping recompilation. This command ensures the contracts are ready for deployment and can be seamlessly integrated into the development workflow for Ethereum-based decentralized applications.



```
Compiling your contracts...
===========================
> Compiling .\contracts\VehicleDetails.sol
> Artifacts written to C:\Users\Asus\Desktop\vehicle-blockchain\build\contracts
> Compiled successfully using:
   - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang
```

Figure 4.2: Contract compilation

- **Truffle Migrate:** The command truffle migrate is used to deploy smart contracts to a blockchain network. It executes the deployment scripts located in the migrations/ directory, which define how and where the contracts should be deployed. During execution, Truffle keeps track of previously deployed migrations in a special migration contract, ensuring that only new or updated contracts are redeployed. The deployment process can be configured to target specific networks by specifying them in the truffle-config.js file.In truffle-config.js, Networks section configures the blockchain network to which you want to deploy the smart contracts. In this case, it specifies a local network that is Ganache running on 127.0.0.1:7545 and Compiler section specifies the version of the Solidity compiler (solc) to use. Here, it’s set to version 0.8.0, which should match the version used in the contracts.



```
Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


2_deploy_vehicle_details.js
===========================

   Replacing 'VehicleDetails'
   --------------------------
   > transaction hash:    0x7b136b28d00eca327c33e529e1576f14b560f13430f4bbe672d2610f7996c7a7
   > Blocks: 0            Seconds: 0
   > contract address:    0x1B6078639F28AcAcf17AF6a815765A812d2F1E99
   > block number:        1
   > block timestamp:     1735531765
   > account:             0x335d46E4AB73d2D761871026B20DE288C2a4074A
   > balance:             99.995970898
   > gas used:            1193808 (0x123750)
   > gas price:           3.375 gwei
   > value sent:          0 ETH
   > total cost:          0.004029102 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:         0.004029102 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.004029102 ETH
```

Figure 4.3: Migration

Transaction hash, blocks created, contract address, block number, block timestamp, account, balance, gas used, gas price, value sent and total cost was obtained.

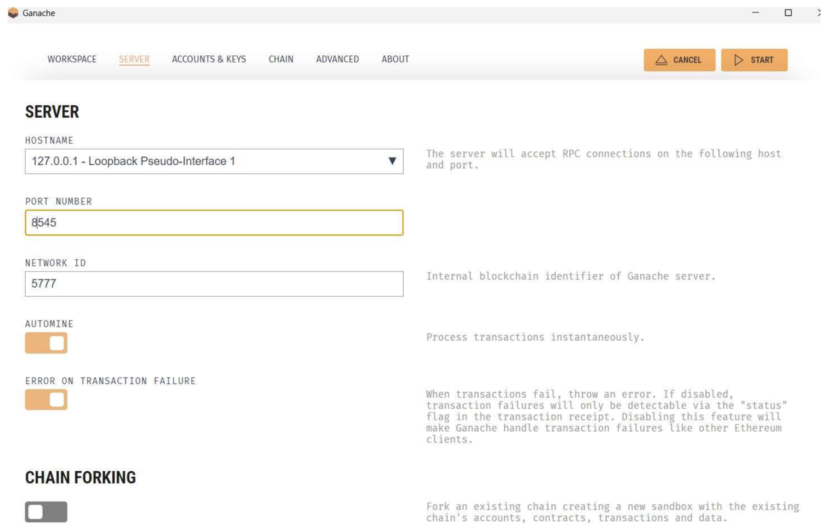
- **Web3 and Metamask integration:** Integrating Web3.js with MetaMask allows users to interact with Ethereum smart contracts directly from their browser. First, ensure that MetaMask is installed and then initialize Web3.js using the window.ethereum provider injected by MetaMask. The user must grant access to their MetaMask account using the ethrequestAccounts method. Once connected, Web3 contract instance can be created using the ABI and address of your deployed contract. With this connection, the contract can interact by calling its methods, such as using call() to read data or send() to send transactions. MetaMask will automatically handle transaction signing and gas fees, prompting the user to approve transactions. In addition to basic interaction, changes to the user's account or network can be listened using the accountsChanged and chainChanged events. This allows the application to dynamically respond when users switch accounts or networks.

4.1.2 Vehicle Insurance Blockchain

- **Truffle Initialization:**To begin, initialize a Truffle project by running the truffle init command. This command creates a project structure consisting of several important directories and files. The contracts/ directory is where InsuranceDetails.sol is stored. The migrations/ directory holds deployment scripts for deploying contracts to a blockchain network. Additionally, the truffle-config.js file is created for configuring network settings, compilers and other options.

- **Key Generation:** Ganache, a personal blockchain for Ethereum development, is used as the blockchain network for the Vehicle

Insurance application. Configured to run on port 8545, Ganache generates multiple Ethereum accounts by default.



SERVER

HOSTNAME
127.0.0.1 - Loopback Pseudo-Interface 1
The server will accept RPC connections on the following host and port.

PORT NUMBER
8545

NETWORK ID
5777
Internal blockchain identifier of Ganache server.

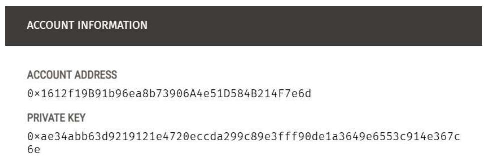
AUTOMINE
Process transactions instantaneously.

ERROR ON TRANSACTION FAILURE
When transactions fail, throw an error. If disabled, transaction failures will only be detectable via the "status" flag in the transaction receipt. Disabling this feature will make Ganache handle transaction failures like other Ethereum clients.

CHAIN FORKING
Fork an existing chain creating a new sandbox with the existing chain's accounts, contracts, transactions and data.

Figure 4.4: Ganache configuration

Account comes with a private key ”0xae34abb3d9219121e4720eccda299c89e3fff90de1a3649e6553c914e367c6e” and the corresponding public address ”0x1612f19B91b96ea8b73906A4e51D584B214F7e6d”.


ACCOUNT INFORMATION

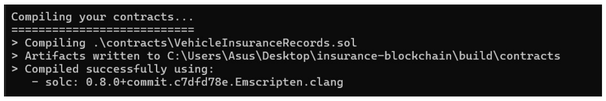
ACCOUNT ADDRESS
0×1612f19B91b96ea8b73906A4e51D584B214F7e6d

PRIVATE KEY
0×ae34abb63d9219121e4720eccda299c89e3fff90de1a3649e6553c914e367c
6e

Figure 4.5: Generated key for Vehicle Insurance Blockchain

- **Compiling Smart Contracts:** The next step is to compile the Solidity smart contracts using the truffle compile command. This command scans the contracts/ directory for all Solidity files (e.g., VehicleDetails.sol and InsuranceDetails.sol) and compiles them. The compiled output is stored in the build/contracts/ directory as .json artifacts. These artifacts include the ABI (Application

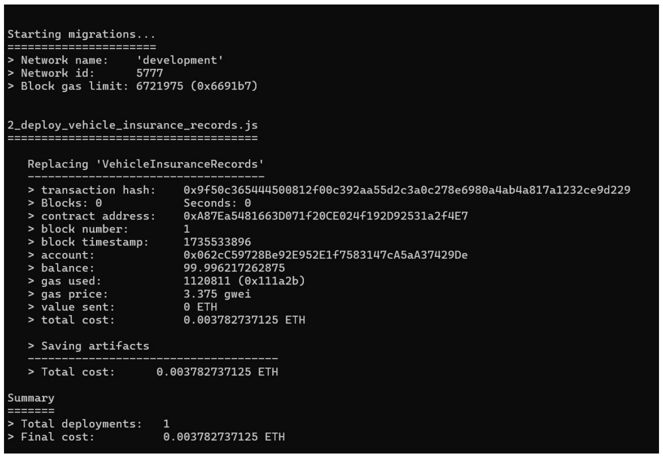
Binary Interface) and bytecode, which are critical for deploying and interacting with the contracts. If no changes are detected in the contracts, Truffle skips recompilation to optimize the process.



```
Compiling your contracts...
===========================
> Compiling .\contracts\VehicleInsuranceRecords.sol
> Artifacts written to C:\Users\Asus\Desktop\insurance-blockchain\build\contracts
> Compiled successfully using:
   - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang
```

Figure 4.6: Compiling Vehicle Insurance smart contract

- **Deploying Smart Contracts:**After the compilation, deploying the smart contracts to the Ganache blockchain using the truffle migrate command. The deployment scripts in the migrations/ directory define the deployment process. In the truffle-config.js file, configure the development network to connect to Ganache at 127.0.0.1:8545.



```
Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


2_deploy_vehicle_insurance_records.js
=====================================

   Replacing 'VehicleInsuranceRecords'
   -----------------------------------
   > transaction hash:    0x9f50c365444500812f00c392aa55d2c3a0c278e6980a4ab4a817a1232ce9d229
   > Blocks: 0            Seconds: 0
   > contract address:    0xA87Ea5481663D071f20CE024f192D92531a2f4E7
   > block number:        1
   > block timestamp:     1735533896
   > account:             0x062cC59728Be92E952E1f7583147cA5aA37429De
   > balance:             99.996217262875
   > gas used:            1120811 (0x111a2b)
   > gas price:           3.375 gwei
   > value sent:          0 ETH
   > total cost:          0.003782737125 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:      0.003782737125 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.003782737125 ETH
```

Figure 4.7: Smart contract deployment

- **Metamask and web3 connection:**Wallet functionality was integrated using MetaMask and Web3.js to streamline user interactions. MetaMask was configured as the primary wallet for signing transactions, while Web3.js facilitated connectivity between the frontend application and the blockchain.

4.2 ORACLE IMPLEMENTATION

4.2.1 Chainlink node installation

Installation includes downloading the official Chainlink node repository to your local machine. The repository contains all the necessary files to run a Chainlink node. Change the current directory to the newly downloaded chainlink folder, where the configuration and deployment scripts are located.

Create a .env file in the chainlink directory with the following content:

- **ROOT:** The root directory for the Chainlink application inside the Docker container.
- **LOG_LEVEL:** Specifies the logging verbosity. info is a balanced choice for general operation.
- **ETH_CHAIN_ID:** Identifies the blockchain network to connect to.
- **LINK_CONTRACT_ADDRESS:** The address of the LINK token contract on the specified network. This address must match the blockchain network being used.
- **CHAINLINK_PORT:** The port number the Chainlink node will use for its API interface (default is 6688).
- **ETH_URL:** The URL of the Ethereum node (or other blockchain nodes).

docker-compose up: This command launches the Chainlink node using Docker Compose, which reads the docker-compose.yml file in the repository to set up and starts chainlink nodes.

4.2.2 Oracle smart contract deployment

For the implementation of oracle, deploy the oracle smart contract. In the Chainlink node, configure bridge tasks that fetch data from one chain and relay it to the other.Fund the deployed oracle contracts with LINK tokens to ensure they can pay for requests.

Algorithm 4.1 Chainlink Oracle Contract

```
1: Initialize Variables:
2:     oracle: Address of the Chainlink oracle
3:     jobId: Unique identifier for the oracle job
4:     fee: Fee in LINK tokens for the oracle request
5:     data: Stores the fetched data from the oracle
6: procedure CONSTRUCTOR
7:     Set public Chainlink token
8:     Initialize oracle with oracle address
9:     Initialize jobId with the job identifier
10:    Initialize fee (0.1 LINK tokens)
11: end procedure
12: procedure REQUESTDATA
13:    Create a Chainlink request object
14:    Add vehicle insurance contract address: endpoint to fetch data
15:    Add path: Path to extract specific data (JSON key)
16:    Send request to the oracle with the specified fee
17: end procedure
18: procedure FULFILL(requestId, fetchedData)
19:    Update data with fetchedData
20:    Call the function on Vehicle Details contract to update with
       fetchedData
21:    Emit RequestFulfilled event with requestId and fetchedData
22: end procedure
23: procedure WITHDRAWLINK
24:    Transfer unused LINK tokens to the owner of the contract
25:    Require successful transfer; otherwise, revert
26: end procedure
```

Cross-domain communication was achieved using Chainlink oracles. These requests were transmitted via Chainlink to the target domain, where the authentication data was validated in the target domain's smart contract. The responses were transmitted back to the initiating domain through the oracle.

CHAPTER 5

RESULTS AND ANALYSIS

5.1 VEHICLE DETAILS USER REGISTRATION PAGE

Vehicle Details Blockchain user registration securely records and manages vehicle owner details. It includes fields for essential information such as the vehicle ID, owner's name, age, gender, contact number and vehicle registration number that ensure each record is uniquely identifiable. Additionally, the tabs for ”Add Owner” and ”Retrieve Details” suggest functionality for both registering new users and accessing existing records.

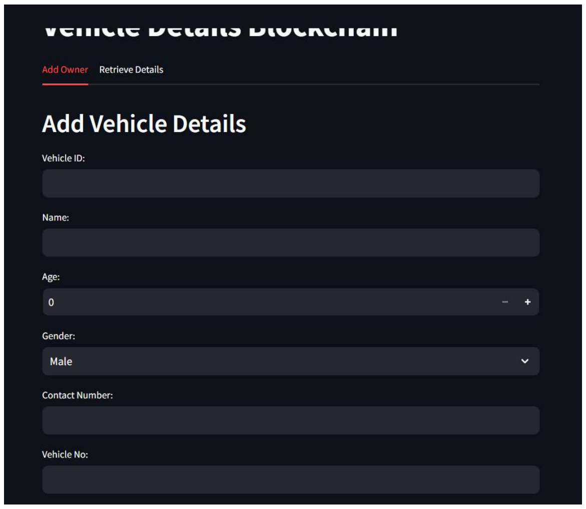


Figure 5.1: Vehicle Details user registration page

5.2 METAMASK TRANSACTION REQUEST FOR VEHICLE DETAILS REGISTRATION

MetaMask transaction request prompt is a part of the process for authorizing blockchain transactions. It displays details such as the source of the request, wallet address and the associated network fee (gas fee) for the transaction. It allows users to review, approve, or reject transactions securely from their wallet. The ”Confirm” button finalizes the transaction, while ”Cancel” aborts it.

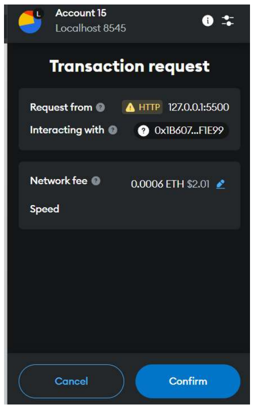
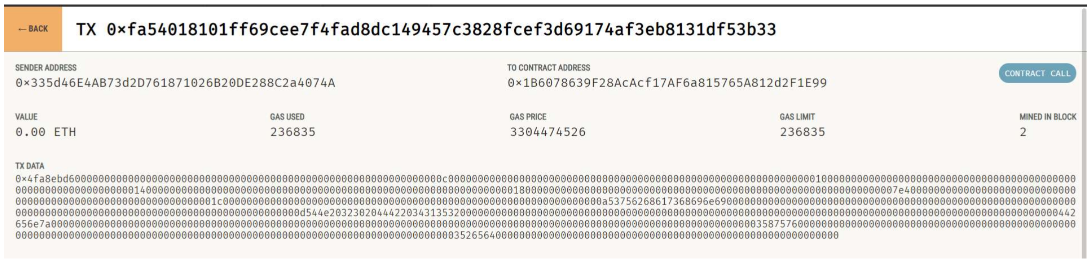


Figure 5.2: Metamask transaction request for Vehicle Details

5.3 TRANSACTION DETAILS IN GANACHE FOR VEHICLE DETAILS BLOCKCHAIN

Details of a transaction included in a mined block on Ganache, with the transaction hash. It includes the Sender Address, which initiated the transaction, and the Contract Address, representing the deployed smart contract being interacted with. The Gas Used and the Gas Price determines the transaction's processing cost. The transaction was mined in block 2 and used a gas limit of 236835 and gas price of 3304474526.



TX 0×fa54018101ff69cee7f4fad8dc149457c3828fcef3d69174af3eb8131df53b33

SENDER ADDRESS	TO CONTRACT ADDRESS
0×335d46E4AB73d2D761871026B20DE288C2a4074A	0×1B6078639F28AcAcf17AF6a815765A812d2F1E99

VALUE	GAS USED	GAS PRICE	GAS LIMIT	MINED IN BLOCK
0.00 ETH	236835	3304474526	236835	2

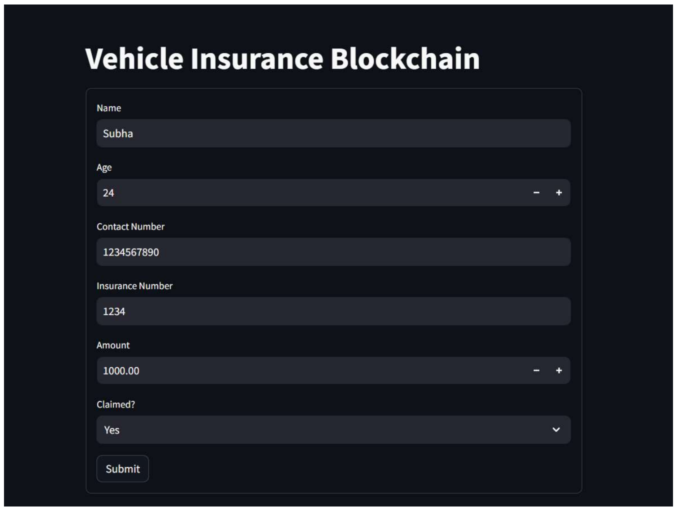
TX DATA

```
0×4fa8ebd600000000000000000000000000000000000000000000000000000000000000c000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000
0000000000000000000014000000000000000000000000000000000000000000000000000000000000001800000000000000000000000000000000000000000000000000000000000007e40000000000000000000000000000
0000000000000000000000000000000001c00000000000000000000000000000000000000000000000000000000000000000a53756268617368696e6900000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000d544e2032302044422034313532000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000442
656e7a00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000035875760000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000003526564000000000000000000000000000000000000000000000000000000000
```

Figure 5.3: Transaction details in ganache for vehicle details

5.4 VEHICLE INSURANCE USER REGISTRATION PAGE

vehicle insurance user registration page designed for a blockchain-based application. Users can enter their personal and insurance details, such as their name, age, contact number, insurance number, insurance amount and claim status. The form ensures proper documentation of key details while maintaining clarity and ease of use. Once submitted, the data is recorded on the blockchain, ensuring secure, transparent and immutable storage of vehicle insurance information, which can be retrieved and verified later.

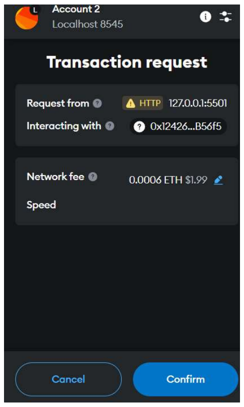

Vehicle Insurance Blockchain

Name	Subha
Age	24
Contact Number	1234567890
Insurance Number	1234
Amount	1000.00
Claimed?	Yes

Figure 5.4: Vehicle Insurance user registration page

5.5 METAMASK TRANSACTION REQUEST FOR VEHICLE INSURANCE REGISTRATION

Transaction request from MetaMask gives details about the interaction between a user's account and a specific smart contract address running on a local blockchain network (localhost:8545). The user is required to pay a network fee to process the transaction, which represents the cost of computational resources. The interface provides options to confirm or cancel the transaction, ensuring user consent before proceeding. This step ensures transparency and security in blockchain interactions


Account 2
Localhost 8545

Transaction request

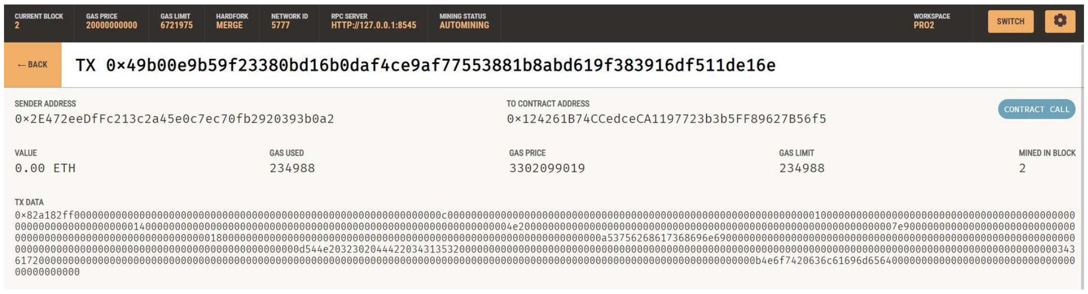
Request from HTTP 127.0.0.1:5501
Interacting with 0x12426...B56f5

Network fee 0.0006 ETH $1.99
Speed

Figure 5.5: Metamask transaction request for Vehicle Insurance

5.6 TRANSACTION DETAILS IN GANACHE FOR VEHICLE INSURANCE BLOCKCHAIN

Blockchain transaction overview displayed on Ganache shows details of a transaction identified by its hash (TX) in the Ethereum network with a network ID of 5777. The transaction used a gas limit of 234,988 and a gas price of 3302099019. It was mined in block 2 and the data section includes the raw transaction input data which is encoded information or instructions for the smart contract.


CURRENT BLOCK 2 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:8545 | MINING STATUS AUTOMINING | WORKSPACE PRO2

TX 0×49b00e9b59f23380bd16b0daf4ce9af77553881b8abd619f383916df511de16e

SENDER ADDRESS
0×2E472eeDfFc213c2a45e0c7ec70fb2920393b0a2

TO CONTRACT ADDRESS
0×124261B74CCedceCA1197723b3b5FF89627B56f5

VALUE
0.00 ETH

GAS USED
234988

GAS PRICE
3302099019

GAS LIMIT
234988

MINED IN BLOCK
2

TX DATA
```
0×82a182ff00000000000000000000000000000000000000000000000000000000000000c00000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000000000000000000
000000000000000000001400000000000000000000000000000000000000000000000000000000000004e200000000000000000000000000000000000000000000000000000000000007e90000000000000000000000000000
000000000000000000000000000000000180000000000000000000000000000000000000000000000000000000000000000a53756268617368696e6900000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000d544e20323020444220343135320000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000343
6172000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000b4e6f7420636c61696d65640000000000000000000000000000000
00000000000
```

Figure 5.6: Transaction details in ganache for vehicle insurance

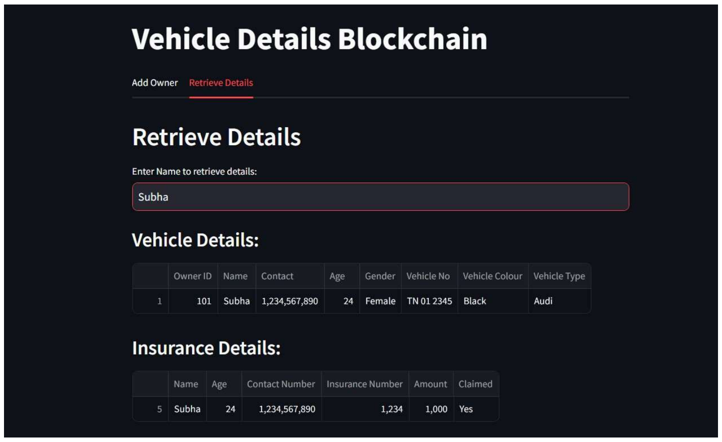
5.7 FETCHING DATA AND DISPLAYED USING ORACLE

Figure 5.7 represents a blockchain-based application for managing and retrieving vehicle and insurance details. It demonstrates the use of an oracle, which is a service that fetches data from the insurance blockchain and integrates it into the vehicle details blockchain.

In this instance, the user queries the system by entering the user name to retrieve details.

- **Vehicle Details:** Information from the vehicle blockchain, including the owner ID, name, contact number, age, gender, vehicle number, vehicle color and vehicle type.
- **Insurance Details:** Data retrieved via the oracle from the insurance blockchain, which includes the name, age, contact number, insurance number, amount and claim status.

This integration highlights cross-chain communication where data consistency, security and accessibility are ensured between separate blockchain networks.



Vehicle Details Blockchain

Retrieve Details

Enter Name to retrieve details: Subha

Vehicle Details:

	Owner ID	Name	Contact	Age	Gender	Vehicle No	Vehicle Colour	Vehicle Type
1	101	Subha	1,234,567,890	24	Female	TN 01 2345	Black	Audi

Insurance Details:

	Name	Age	Contact Number	Insurance Number	Amount	Claimed
5	Subha	24	1,234,567,890	1,234	1,000	Yes

Figure 5.7: Data fetched and displayed using oracle

5.8 PERFORMANCE ANALYSIS

Table 5.1 compares the performance of various schemes ([2], [3], [5], [7], [8]) and the proposed system across six parameters: key agreement, mutual authentication, identity protection, on-demand revocation, domain scalability and deployment cost. The time cost for one user across three systems: Xauth, Hybrid Blockchain and proposed system. Among the systems, Hybrid Blockchain demonstrates the lowest time cost at 50 ms, indicating higher efficiency. While all schemes, including the proposed system, support key agreement and mutual authentication, only the proposed system excels in providing identity protection, on-demand revocation and domain scalability, which are critical for ensuring privacy, security and adaptability in modern environments.

Table 5.1: Comparisons of Functionalities of Different Systems

Scheme	**[2]**	**[3]**	**[5]**	**[7]**	**[8]**	**Proposed System**
Key agreement	Yes	Yes	Yes	Yes	Yes	Yes
Mutual authentication	Yes	Yes	Yes	Yes	Yes	Yes
Identity protection	No	No	Yes	Yes	No	Yes
On demand revocation	Yes	No	Yes	No	No	Yes
Domain scalability	No	No	No	No	No	Yes
Deployment cost	Yes	Yes	Yes	Yes	Yes	No

The time cost for one user across three systems: Xauth, Hybrid Blockchain and proposed system. Among the systems, Hybrid Blockchain demonstrates the lowest time cost at 50 ms, indicating higher efficiency.

5.9 CHALLENGES IN IMPLEMENTATION

Cross-domain authentication using oracles facilitates secure and reliable data transfer between blockchain networks but introduced several challenges during implementation. Large and interdependent smart contracts can result in longer compilation times and increased difficulty in debugging errors, complicating the development process. Additionally, inaccurate gas limit estimation during deployment can lead to out-of-gas errors or failed transactions, further hindering progress. Configuring the Chainlink node adds another layer of complexity, as meticulous attention is required when setting up .env variables or Docker containers, with even minor errors potentially preventing the node from functioning. Cross-domain communication via oracles introduces latency, which can delay updates to smart contracts, particularly in time-sensitive applications. Moreover, robust error handling is critical to ensure the system can gracefully manage failures in oracle responses, as disruptions in fetching or transmitting data could significantly impact the application's functionality.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

This implementation demonstrates the feasibility and efficiency of leveraging blockchain technologies, decentralized oracles, and Web3 frameworks to achieve secure, privacy-preserving cross-domain authentication. The proposed system introduces a scalable approach using decentralized oracle systems (DOS) to facilitate communication between on-chain and off-chain environments while maintaining data integrity and security. By employing zero-knowledge proof, the system ensures user confidentiality and prevents identity leakage during the authentication process.

While the existing systems support secure communication and mutual trust, the Proposed System, ensure user confidentiality and dynamic access control. The Proposed System stands out in terms of identity protection, on-demand revocation, and domain scalability, which are critical for modern, scalable and secure cross-domain authentication.

Unlike traditional consortium blockchain-based models, this solution achieves cost efficiency and scalability by eliminating the need for a shared consortium blockchain, instead allowing each domain to maintain its local blockchain. The use of Chainlink as the oracle network strengthens the overall infrastructure by securely transferring data across domains. Furthermore, the system's robust architecture provides resilience against various attacks, including impersonation, forgery, and man-in-the-middle attacks, while maintaining efficiency in computational and gas costs. Overall, this implementation serves as a significant advancement in blockchain-based cross-domain authentication, addressing critical challenges in privacy, scalability, and deployment costs, making it well-suited for real-world applications.

6.2 FUTURE WORK

Oracle-based Cross Domain Authentication’s future work would be based on acquiring the aspects of its performance and security. One key focus will be optimizing computational and gas costs during smart contract execution to improve overall system efficiency. The exploration of alternative decentralized oracle solutions with faster response times and reduced fees will also be a priority. Enhancing user anonymity through advanced cryptographic techniques, such as advanced proof generation algorithms, could further fortify privacy protections. Additionally, permissioned blockchains like Hyperledger could be integrated to provide more controlled and secure environments for sensitive operations. These advancements aim to establish the system as a benchmark for secure, privacy-preserving authentication frameworks in blockchain ecosystems.

REFERENCES

[1] Meng Shen, Huisen Liu, Liehuang Zhu, Ke Xu, Hongbo Yu, Xiaojiang Du, and Mohsen Guizani. Blockchain-assisted secure device authentication for cross-domain industrial iot. *IEEE Journal on Selected Areas in Communications*, 38(5):942–954, 2020.

[2] Jing Chen, Zeyi Zhan, Kun He, Ruiying Du, Donghui Wang, and Fei Liu. Xauth: Efficient privacy-preserving cross-domain authentication. *IEEE Transactions on Dependable and Secure Computing*, 19(5):3301–3311, 2021.

[3] NS Akhilesh, MN Aniruddha, and KS Sowmya. Implementation of blockchain for secure bank transactions. In *2020 International Conference on Mainstreaming Block Chain Implementation (ICOMBI)*, pages 1–10. IEEE, 2020.

[4] Sandeep Srivastava, Deepshikha Agarwal, and Brijesh Chaurasia. Secure decentralized identity management using blockchain. In *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1355–1360. IEEE, 2023.

[5] Yong Yuan and Fei-Yue Wang. Blockchain and cryptocurrencies: Model, techniques, and applications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(9):1421–1428, 2018.

[6] Asma A Alhussayen, Kamal Jambi, Maher Khemakhem, and Fathy E Eassa. A blockchain oracle interoperability technique for permissioned blockchain. *IEEE Access*, 2024.

[7] Wei Tong, Ce Shen, Zesong Dong, and Jian Li. Interoperability solution for blockchain-based internet of vehicles driven by multiple oracles. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pages 196–202. IEEE, 2023.

[8] Chaosheng Feng, Bin Liu, Zhen Guo, Keping Yu, Zhiguang Qin, and Kim-Kwang Raymond Choo. Blockchain-based cross-domain authentication for intelligent 5g-enabled internet of drones. *IEEE Internet of Things Journal*, 9(8):6224–6238, 2021.

[9] Zhihua Cui, XUE Fei, Shiqiang Zhang, Xingjuan Cai, Yang Cao, Wensheng Zhang, and Jinjun Chen. A hybrid blockchain-based identity authentication scheme for multi-wsn. *IEEE Transactions on Services Computing*, 13(2):241–251, 2020.

[10] Miaomiao Wang, Lanlan Rui, Yang Yang, Zhipeng Gao, and Xingyu Chen. A blockchain-based multi-ca cross-domain authentication scheme in decentralized autonomous network. *IEEE Transactions on Network and Service Management*, 19(3):2664–2676, 2022.

[11] Lingyan Xue, Haiping Huang, Fu Xiao, and Wenming Wang. A cross-domain authentication scheme based on cooperative blockchains functioning with revocation for medical consortiums. *IEEE Transactions on Network and Service Management*, 19(3):2409–2420, 2022.

[12] Debiao He, Neeraj Kumar, Huaqun Wang, Lina Wang, Kim-Kwang Raymond Choo, and Alexey Vinel. A provably-secure cross-domain handshake scheme with symptoms-matching for mobile healthcare social network. *IEEE Transactions on Dependable and Secure Computing*, 15(4):633–645, 2016.

[13] Baobao Chai, Jiguo Yu, Biwei Yan, Yong Yu, and Shengling Wang. Bscda: Blockchain-based secure cross-domain data access scheme for internet of things. *IEEE Transactions on Network and Service Management*, 2024.

[14] I. Bashir. *Mastering Blockchain: Distributed ledger technology, decentralization, and smart contracts explained, 2nd Edition*. Packt Publishing, 2018.