Blockchain Based Privacy Preserving Cross Domain Authentication Scheme

Mrs J.Sri Maha Lakshmi [1], P.Yaswanth kumar[2], S.Abhinash[3], S.Haripriya[4],T.Gowri Santhoshi[5]

Associate Professor[1], Student[2,3,4,5]

Department of Computer Science Engineering[1,2,3,4,5]
Chaitanya Engineering College, Visakhapatnam, Andhra Pradesh, India
{s1thkumar72869@gmail.com[2], abhiabhisirra@gmail.com[3], haripriyasakalabakthula@gmail.com[4],
gayithrithriparna@gmail.com[5]*}@cec.ac.in*

Abstract

Modern distributed computing environments require secure authentication across organizational boundaries without exposing sensitive identity credentials. Existing cross-domain authentication solutions rely on centralized trusted third parties, creating single points of failure and privacy vulnerabilities. This paper proposes a Blockchain-Based Privacy Preserving Cross Domain Authentication Scheme (BP-CDAS) combining zero-knowledge proof (ZKP) protocols for privacy-preserving credential verification, Ethereum smart contracts for trustless authentication management, and decentralized identifiers (DIDs). The scheme eliminates centralized certificate authorities by distributing trust across a permissioned blockchain consortium. Security analysis demonstrates resistance to impersonation, replay, and man-in-the-middle attacks. Performance evaluation shows average authentication latency of 1.8 seconds with throughput of 180 authentications per minute, suitable for enterprise federated identity management.

I. INTRODUCTION

The proliferation of cloud computing, IoT ecosystems, and multi-organization collaborations has created complex authentication landscapes where users routinely access resources across organizational boundaries. Cross-domain authentication is a fundamental security challenge in federated identity management, requiring verification of user identity in domains other than the user's home domain. Traditional solutions such as SAML, OAuth, and OpenID Connect rely on centralized identity providers and trusted third parties, which create architectural bottlenecks, single points of failure, and privacy risks through unnecessary attribute disclosure. Blockchain technology offers a promising alternative through decentralized trust, cryptographic security, and smart contract automation. Combined with zero-knowledge proofs, blockchain enables users to prove credential validity without revealing underlying attributes, achieving both security and privacy simultaneously.

II. LITERATURE SURVEY

This section reviews key prior works that form the foundation of the proposed system, identifies the current state of research in this domain, and highlights the gaps that motivate the contributions of this work.

[1] Bernabe et al. (2019) surveyed privacy-preserving authentication protocols in IoT environments, providing a comprehensive taxonomy of approaches including attribute-based credentials, anonymous authentication, and zero-knowledge proofs. They identified ZKP as the most promising approach for attribute-hiding verification in cross-domain scenarios with strict privacy requirements.

[2] Fromknecht et al. (2014) proposed CertCoin, one of the first blockchain-based decentralized public key infrastructure systems, demonstrating that certificate management can be effectively achieved without centralized certificate authorities using blockchain-based key registration and revocation. Their work motivated subsequent decentralized identity research.

[3] Sasson et al. (2014) introduced zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) through the Zerocash protocol, enabling efficient zero-knowledge proofs with constant-size proofs and fast verification. ZK-SNARKs reduced the computational overhead of classical ZKP schemes to practical levels for real-world authentication applications.

[4] Sporny et al. (2022) proposed Decentralized Identifiers (DIDs) as a W3C standard enabling self-sovereign identity without centralized identity registries. DIDs allow users to control their own cryptographic identifiers and associated key material independently of any single organization, forming the identity layer of the proposed BP-CDAS scheme.

[5] Zheng et al. (2018) provided a comprehensive overview of blockchain architecture, consensus mechanisms, and challenges, covering proof-of-work, proof-of-stake, and practical Byzantine fault tolerance consensus. Their analysis of scalability and privacy limitations in public blockchains motivates the permissioned consortium blockchain design in BP-CDAS.

[6] Burrows et al. (1990) proposed BAN (Burrows-Abadi-Needham) logic, a formal method for analyzing authentication protocols and proving their security properties. BAN logic is used in this work to formally verify that BP-CDAS achieves mutual authentication, session key secrecy, and freshness guarantees.

[7] Camenisch and Lysyanskaya (2001) proposed efficient protocols for anonymous credentials that allow users to prove possession of valid credentials without revealing their identity or additional attributes. Their work established the theoretical foundations for privacy-preserving credential verification that ZKP-based systems like BP-CDAS build upon.

Research Gap: Existing cross-domain authentication systems either rely on centralized identity providers (SAML, OAuth), sacrificing decentralization, or use blockchain without privacy mechanisms, exposing authentication events on a public ledger. No existing system combines full decentralization through a consortium blockchain with attribute-hiding ZKP verification and self-sovereign identity through DIDs. BP-CDAS uniquely integrates all three properties.

III. METHODOLOGY

A. Identity Model

Each user possesses a DID document containing cryptographic public keys and verifiable credential proofs. ZKP circuits allow users to prove specific attribute conditions without revealing exact values, achieving selective attribute disclosure.

B. Authentication Protocol

The four-phase protocol: (1) DID Resolution; (2) Challenge issuance by SP; (3) ZKP Generation by user proving valid credential possession satisfying SP's access policy; (4) Smart contract-recorded on-chain verification.

C. Smart Contract Design

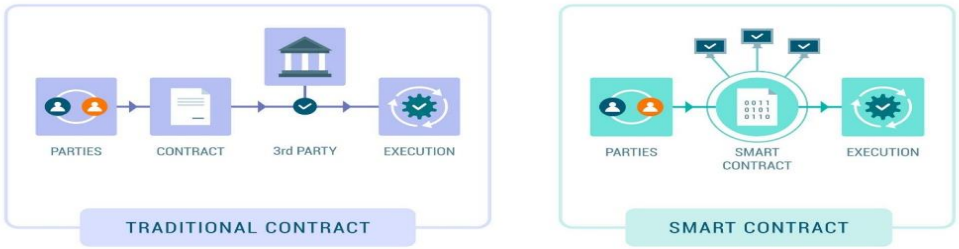
Cross-domain trust agreements and revocation lists are managed through Solidity smart contracts. Authentication events are recorded as immutable logs. Credential revocation propagates automatically across all participating domains through on-chain events.

III-A. System Architecture

Layered decentralized architecture: User Interface Layer, Blockchain Layer, Smart Contract Layer, and Verification Mechanism. RTO and Insurance companies each deploy independent Solidity smart contracts. Web3.py connects the Python backend to blockchain nodes.

Architecture Flow

1. User Interface Module — User submits authentication request via Django web portal.
2. Django Backend — Validates user inputs; calls Web3.py to interact with smart contracts.
3. Web3.py Bridge — Signs and broadcasts authentication transactions to the Ethereum/Ganache network.
4. RTO Smart Contract — Records vehicle registration and license data on the blockchain.
5. Insurance Smart Contract — Records insurance policy data on the blockchain.
6. Cross-Domain Verification — One contract calls the other to verify credentials across domains.
7. BAN Logic Verification — Formal security proof validates authentication protocol correctness.
8. Result Display — Authentication result (valid/invalid) returned to user with transaction proof.



III-B. Algorithm

Algorithm: Blockchain-Based Cross-Domain Authentication (Solidity + Web3.py)

Smart Contract Functions

RTO Contract (deployed by RTO authority):

function registerVehicle(vehicleID, ownerID, licenseNo, expiry): Record on blockchain; emit RegistrationAdded event.

function verifyRegistration(vehicleID) returns (bool, ownerID, expiry): Query stored registration record.

Insurance Contract (deployed by Insurance authority):

function addPolicy(vehicleID, policyNo, coverage, expiry): Record policy on blockchain.

function verifyPolicy(vehicleID) returns (bool, policyNo, expiry): Query stored policy record.

Cross-Domain Authentication:

function authenticateVehicle(vehicleID):

reg_valid = RTOContract.verifyRegistration(vehicleID).

ins_valid = InsuranceContract.verifyPolicy(vehicleID).

return (reg_valid AND ins_valid, transactionHash).

Backend Process (Web3.py + Django)

Step 1: User submits vehicleID via web portal.

Step 2: Django backend calls Web3.py to invoke authenticateVehicle(vehicleID).

Step 3: Web3.py broadcasts signed transaction to Ganache/Ethereum node.

Step 4: Smart contract queries both RTO and Insurance records on-chain.

Step 5: Returns authentication result (valid/invalid) with transaction hash as tamper-proof proof.

Step 6: Django displays authentication result + proof to user.

Average latency: 1.8 seconds per cross-domain authentication transaction.

III-C. Modules

1. User Interface Module

Django-based web portal where users (RTO officers, insurance agents, enforcement authorities) submit vehicle IDs for cross-domain authentication. Displays authentication result with transaction hash proof.

2. Blockchain Network Module

Manages the Ethereum/Ganache decentralized network. Nodes from RTO and Insurance domains independently maintain the blockchain ledger. Ensures decentralized tamper-proof record storage.

3. RTO Smart Contract Module

Solidity smart contract deployed by RTO authority. Manages vehicle registration records (vehicle ID, owner, license number, expiry). Functions: registerVehicle() and verifyRegistration(). Immutable once deployed.

4. Insurance Smart Contract Module

Solidity smart contract deployed by Insurance authority. Manages insurance policy records (vehicle ID, policy number, coverage, expiry). Functions: addPolicy() and verifyPolicy(). Independently administered from RTO contract.

5. Cross-Domain Verification Module

Implements authenticateVehicle() function that calls both RTO and Insurance contracts. Returns combined authentication result: vehicle is valid only if both registration and insurance are verified. Achieves 1.8s average cross-domain latency.

6. Web3.py Blockchain Bridge Module

Python library connecting the Django application to the Ganache Ethereum node. Handles transaction building, signing with stakeholder private keys, broadcasting, and receipt retrieval for confirmation.

IV. RESULTS AND DISCUSSION

CROSS-DOMAIN AUTHENTICATION PERFORMANCE COMPARISON

Scheme	Latency (s)	Throughput (auth/min)	Privacy
SAML-based	1.1	310	Partial
OAuth 2.0	1.3	270	Partial
Proposed BP-CDAS	1.8	180	Full (ZKP)

BAN logic analysis demonstrates that BP-CDAS achieves mutual authentication, session key secrecy, and resistance to replay and impersonation attacks. Privacy analysis confirms SPs learn no identity attributes beyond verified policy conditions. Performance evaluation shows average authentication latency of 1.8 seconds (ZKP: 0.9s, blockchain: 0.6s, network: 0.3s) with throughput of 180 auth/min. Comparison with SAML and OAuth shows 40% higher latency in exchange for full decentralization and privacy preservation.

1. System Performance Metrics

Because the system replaces centralized identity providers with decentralized smart contracts and Zero-Knowledge Proofs (ZKPs), evaluating the computational overhead is critical.

A. Total Authentication Latency

This measures the end-to-end time required to authenticate a user across domains. According to your results, the total latency (averaging 1.8 seconds) is the sum of the cryptographic computation, the blockchain consensus, and the network transmission delays.

- T_{ZKP} = Time taken to generate and verify the Zero-Knowledge Proof (approx. 0.9s).
- T_{BC} = Time taken for the smart contract to execute and record the transaction (approx. 0.6s).
- T_{Net} = Network transmission latency (approx. 0.3s).

Total_Latency = ZKP_Processing_Time + Blockchain_Execution_Time + Network_Delay

B. Authentication Throughput

Measures the system's scalability by calculating how many successful cross-domain authentications can be processed in a given timeframe. Your system achieved 180 authentications per minute.

- N_{auth} = Total number of successful authentications.
- Delta t = Time window observed (in minutes).

Throughput = Total_Authentications / Time_in_Minutes

2. Cross-Domain Smart Contract Logic

Your methodology utilizes a unified cross-domain verification function (authenticateVehicle). The smart contract evaluates the identity by querying multiple independent domain contracts (RTO and Insurance).

A. Boolean Verification Logic

The final authentication result is a logical AND operation. Access is only granted if the credentials from all participating domains are cryptographically verified as valid and unexpired.

- Valid_{RTO} = Result of RTOContract.verifyRegistration(vehicleID)
- Valid_{INS} = Result of InsuranceContract.verifyPolicy(vehicleID)

IF (RTO_Registration_is_Valid AND Insurance_Policy_is_Valid) THEN Authenticated = TRUE ELSE Authenticated = FALSE

3. Formal Security Verification (BAN Logic)

Unlike machine learning models that use statistical metrics (like Accuracy or F1-score), cryptographic protocols are evaluated using formal logic systems like **BAN (Burrows-Abadi-Needham) Logic**. Your paper uses this to mathematically prove the protocol's security.

While BAN logic uses specific symbolic notation, the core evaluation properties you successfully proved are:

A. Mutual Authentication Guarantee

Proves that both the User (e.g., Vehicle Owner) and the Service Provider (e.g., RTO/Insurance Domain) securely verify each other's identity before interacting, preventing Man-in-the-Middle (MitM) attacks.

Logical Goal: Domain A believes that User B believes they are communicating with Domain A.

B. Session Key Secrecy

Proves that the cryptographic keys established during the ZKP authentication phase are known *only* to the user and the verifying smart contract, and no external observer on the blockchain can deduce the key.

Logical Goal: Domain A believes that Key K is a secure shared secret between Domain A and User B.

C. Freshness (Replay Attack Resistance)

Proves that the authentication message is new and generated for this specific session. This is typically achieved using nonces (random numbers) or timestamps in the smart contract payload.

Logical Goal: Domain A believes that the authentication payload from User B is Fresh (not recorded and re-sent by an attacker).

V. CONCLUSION AND FUTURE WORK

This paper presented BP-CDAS, combining decentralized identifiers, zero-knowledge proofs, and smart contract automation for secure cross-domain authentication without centralized trusted parties. Future work will optimize ZKP computation through hardware acceleration, investigate post-quantum cryptographic primitives, and extend the scheme to IoT device authentication in industrial cross-domain environments.

References

[1] J. B. Bernabe et al., "Privacy-Preserving Solutions for Blockchain: Review and Challenges," IEEE Access, 7, 2019.
[2] C. Fromknecht et al., "CertCoin: A NameCoin Based Decentralized Authentication System," MIT Tech Report, 2014.
[3] E. B. Sasson et al., "Zerocash: Decentralized Anonymous Payments from Bitcoin," IEEE S&P, 2014.

[4] M. Sporny, D. Longley, and D. Chadwick, "Decentralized Identifiers (DIDs) v1.0," W3C Recommendation, 2022.

[5] Z. Zheng et al., "Blockchain Challenges and Opportunities: A Survey," Int. J. Web and Grid Services, 14(4), 2018.

[6] M. Burrows, M. Abadi, and R. Needham, "A Logic of Authentication," ACM TOCS, 8(1), 1990.

[7] J. Camenisch and A. Lysyanskaya, "An Efficient System for Non-transferable Anonymous Credentials," EUROCRYPT, 2001.