Building Cardano Dapp with Web3Auth, MeshSDK and on-chain chat data privacy-preserving with ZKP

Abstract
This white paper outlines a novel approach to integrating the Cardano blockchain with MeshSDK, enabling wallet derivation from Web3Auth wallets, encrypted chat storage on-chain, and keyword search using Zero-Knowledge Proofs (ZKPs) without revealing the complete chat history. By leveraging Merkle Trees for chat message integrity verification and robust encryption methods, this approach enhances privacy and scalability for decentralized communication systems.

1. Introduction

In the age of digital communication, privacy and security have become paramount concerns. Traditional centralized messaging platforms often require users to trust third parties with their sensitive data, making them vulnerable to data breaches, censorship, and unauthorized surveillance. Decentralized messaging systems built on blockchain technology offer a promising alternative by enabling users to communicate securely without relying on a central authority.

Cardano, a third-generation blockchain platform, is uniquely positioned to support decentralized applications with strong security, scalability, and sustainability features. By leveraging Cardano's robust infrastructure, developers can build decentralized chat systems that maintain data integrity and user privacy.

This white paper presents a novel approach to secure, privacy-preserving chat on the Cardano blockchain. It outlines the integration of Web3Auth for seamless user authentication and wallet management, using MeshSDK to derive Cardano wallet addresses from Web3Auth wallets. The system encrypts all user chat data before storing it on the blockchain to ensure confidentiality.

To guarantee data integrity and enable efficient verification, Merkle trees are employed to organize encrypted messages, with Merkle roots anchored on-chain. Further enhancing privacy, the paper explores the use of zero-knowledge proofs (ZKP) to enable keyword search on encrypted chat data without revealing message content, allowing users to query chat history securely.

The goals of this paper are to:

- Provide a clear methodology for integrating Web3Auth with Cardano wallets via MeshSDK.

- Demonstrate how encrypted chat messages can be securely stored and verified using Merkle trees on the Cardano blockchain.

- Explore privacy-preserving search techniques using zero-knowledge proofs, addressing the challenges of querying encrypted data on a public ledger.

By combining these technologies, the proposed system aims to create a decentralized chat platform that ensures user privacy, data security, and verifiable message integrity, paving the way for a new generation of confidential blockchain-based communication tools.

2. Preliminaries and Technologies

This section outlines the key foundational technologies and cryptographic concepts leveraged in building a secure, privacy-preserving decentralized chat system on Cardano. A clear understanding of these components is crucial for grasping the system architecture and design decisions.

2.1 Cardano Blockchain Overview

Cardano is a third-generation blockchain platform developed with a research-driven approach. Its architecture is designed for scalability, security, and sustainability.

- **Ouroboros Proof-of-Stake (PoS) Consensus:** Ouroboros uses stake delegation and randomized slot leader election to validate blocks efficiently and securely, reducing energy consumption compared to proof-of-work systems.

- **Layered Architecture:**

    - **Cardano Settlement Layer (CSL):** Handles ADA transactions and native token transfers with deterministic finality.

    - **Cardano Computation Layer (CCL):** Supports smart contracts written in Plutus (based on Haskell), enabling complex decentralized logic.

- **On-Chain Metadata Storage:** Cardano transactions can carry small payloads of arbitrary metadata (up to ~16KB). This feature is ideal for anchoring cryptographic proofs (like Merkle roots) or storing encrypted message hashes, minimizing on-chain storage costs.

- **UTXO Model:** Cardano extends Bitcoin's UTXO (Unspent Transaction Output) model to a **Extended UTXO (EUTXO)** model, supporting expressive smart contracts and predictable transaction behavior, beneficial for managing encrypted message states.

2.2 MeshSDK

MeshSDK is a comprehensive JavaScript/TypeScript toolkit that abstracts complex Cardano blockchain interactions, providing streamlined development for dApps.

- **Wallet Address Derivation:** MeshSDK supports deriving Cardano wallet addresses from private keys managed externally, including those obtained via Web3Auth. It implements Cardano's hierarchical deterministic (HD) wallet standards (CIP-1852), supporting BIP32-ED25519 key derivation paths.

- **Transaction Building and Signing:** It allows developers to construct complex Cardano transactions, including metadata, multi-asset transfers, and smart contract interactions, and facilitates signing these transactions via connected wallets or key stores.

- **Node Communication:** MeshSDK interacts with Cardano nodes through REST APIs or direct network calls, enabling querying blockchain state, submitting transactions, and listening for events.

2.3 Web3Auth Wallet System

Web3Auth abstracts blockchain wallet complexity by providing users with seamless login experiences backed by secure key management.

- **Key Management:**

    - Users authenticate via social login providers (Google, Facebook, Apple, etc.) or passwordless mechanisms.

    - Web3Auth securely generates and manages private keys, often splitting key shares using threshold cryptography or distributed key generation techniques to reduce single points of failure.

- **Cardano Compatibility:** Web3Auth can generate keys compatible with Cardano's cryptography (Ed25519), enabling direct wallet address derivation and transaction signing within the Cardano ecosystem.

- **SDK Integration:** Through MeshSDK integration, Web3Auth wallets can sign Cardano transactions without exposing private keys to the dApp, preserving user security.

2.4 Cryptographic Primitives

Encryption

- **Symmetric Encryption (AES-GCM):** Used for encrypting chat message content efficiently. AES in Galois/Counter Mode provides confidentiality and integrity with authenticated encryption.

- **Asymmetric Encryption (Elliptic Curve Cryptography - Ed25519 or Curve25519):** Employed for secure key exchange between users to share symmetric keys, enabling end-to-end encrypted chat.

- **Key Management:** Users maintain key pairs derived from Web3Auth wallets, enabling cryptographic operations without exposing raw private keys.

Hashing

- **Cryptographic Hash Functions:** SHA-256 or Blake2b produce fixed-length digests of message data, providing collision resistance and preimage resistance essential for data integrity.

- **Usage in Merkle Trees:** Hashes of encrypted messages form leaf nodes; internal nodes are hashes of concatenated child nodes.

Merkle Trees

- **Structure:** A binary tree where each non-leaf node is the hash of its two child nodes. The root hash represents the integrity of the entire dataset.

- **Efficient Verification:** Merkle proofs (paths from leaf to root) enable verification of individual message inclusion without revealing or processing the entire dataset.

- **On-Chain Anchoring:** The Merkle root is stored as metadata in a Cardano transaction, acting as a tamper-proof commitment to all messages in the tree.

2.5 Zero-Knowledge Proofs (ZKP)

- **Definition:** Protocols allowing a prover to convince a verifier that a statement is true without revealing any underlying information beyond the validity of the statement itself.

- **ZKP for Keyword Search:**

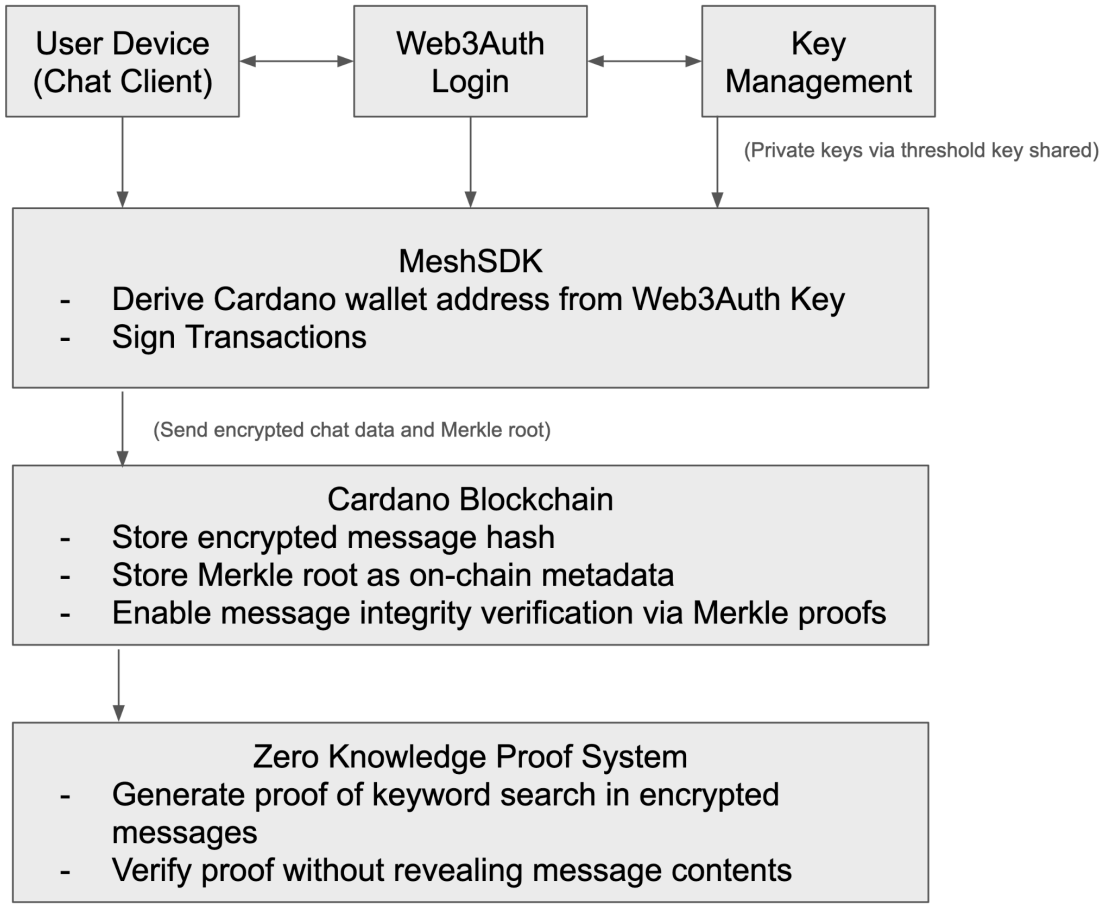
    - The prover demonstrates the existence of a keyword in encrypted chat messages without decrypting or exposing message contents.

    - Approaches include constructing zk-SNARKs or zk-STARKs over commitments derived from encrypted data and Merkle proofs.

- **Integration with Merkle Trees:** ZKPs verify the presence of a keyword in a message corresponding to a leaf node of the Merkle tree anchored on-chain, ensuring integrity and privacy simultaneously.

- **Performance Considerations:** Generating and verifying ZKPs require computational resources; optimizing circuits and leveraging off-chain computation helps maintain usability.

Diagram: System Architecture and Flow



Zero-Knowledge Proof Implementation Guidance for Privacy-Preserving Keyword Search

1. What is the goal?

Allow a user to **prove** that a certain keyword exists in an encrypted chat message stored on-chain (or referenced on-chain) **without revealing the message content or the keyword itself**. The verifier only learns the proof is valid but gains no extra info.

2. Conceptual Approach

- Messages are encrypted and stored off-chain or partially on-chain (e.g., hashes or encrypted blobs).

- Each message corresponds to a leaf in a Merkle tree, with the Merkle root stored on-chain to ensure integrity.

- The prover generates a zero-knowledge proof that they know an encrypted message in the tree whose decrypted plaintext contains the keyword.

- The proof is verified against the Merkle root, proving inclusion and correctness without revealing data.

3. Key Components

Component	Description
Commitments	Cryptographic commitments to encrypted messages and keywords used as inputs to the ZKP circuit.
Merkle Tree Inclusion	Proof that the encrypted message is part of the Merkle tree anchored on-chain (proof of inclusion).
Keyword Matching Circuit	A ZKP circuit that verifies the keyword exists in the decrypted message (usually done by hashing).
Proof System	zk-SNARK, zk-STARK, Bulletproofs, or other proving systems to generate and verify proofs.

4. Choosing a ZKP Framework

- **circom + snarkjs:** Popular for zk-SNARK circuit design and proof generation.

- **ZoKrates:** High-level language for ZKP circuit generation and Solidity integration.

- **Halo2 / Arkworks / gnark:** Rust-based libraries offering zk-SNARKs/STARKs with more control but steeper learning curve.

- **StarkWare:** For zk-STARKs, better scalability but complex setup.

5. High-Level Steps to Build the ZKP System

Step 1: Define the ZKP Circuit

- Inputs:

    - Private: encrypted message ciphertext, decryption key, keyword

    - Public: Merkle root (from blockchain), hash function parameters

- Computation inside circuit:
    - Decrypt the ciphertext with the key (or simulate it by pre-hashing)
    - Check if the keyword exists in the plaintext (could be a substring or hash equality)
    - Compute the Merkle proof from leaf (hashed encrypted message) to root and verify inclusion
- Output:
    - Validity of keyword presence and inclusion proof

Note: Actual decryption inside a circuit is expensive, so often you store commitments or hashes of keywords or use Oblivious RAM (ORAM) or private set membership proofs.

Step 2: Compile and Generate Trusted Setup

- Compile circuit (e.g., with circom) into an arithmetic circuit.
- Perform a trusted setup ceremony if using zk-SNARKs (unless using transparent setups like zk-STARKs).

Step 3: Generate Proofs

- Prover runs the circuit with actual inputs (encrypted message, keyword, Merkle proof).
- Generates a proof that the verifier can check.

Step 4: Verify Proof

- Verifier uses on-chain or off-chain verifier smart contract to verify proof correctness.
- Accept or reject the query based on proof validity.

6. Practical Simplifications

- **Precompute keyword hashes:** Instead of searching substrings in plaintext, store hashes of keywords or tags encrypted alongside messages. The ZKP circuit proves knowledge of a hash match.

- **Use deterministic encryption or searchable encryption schemes** to reduce circuit complexity.

- **Offload expensive computations off-chain**, verifying only succinct proofs on-chain.

7. Integration with Cardano and MeshSDK

- Use MeshSDK to submit proofs or commitments on-chain as metadata or interact with smart contracts verifying proofs.

- Off-chain servers or clients generate proofs and send queries with ZKP to the network or other users.

- Design UI/UX so users can submit keyword queries and receive proof-verified responses without exposing message content.

3. Wallet Integration and Address Derivation

Integrating user authentication with blockchain wallet management is a critical step for building a seamless decentralized chat system. This section details how Web3Auth and MeshSDK collaborate to provide a smooth user experience while maintaining security and compatibility with the Cardano blockchain.

3.1 User Authentication via Web3Auth

Web3Auth simplifies user onboarding by abstracting away traditional private key management through familiar authentication methods such as social logins (Google, Facebook, Twitter) or passwordless authentication. When a user logs in:

- Web3Auth performs authentication with the chosen provider and generates cryptographic key shares, often leveraging threshold cryptography to securely split and store the private key shares.

- The user's private key is reconstructed on the client side without ever transmitting it to a central server, ensuring security and decentralization.

- This private key is compatible with Cardano's cryptographic standards (Ed25519 curve), enabling blockchain operations such as signing transactions.

3.2 Deriving Cardano Wallet Addresses with MeshSDK

MeshSDK facilitates interaction with Cardano's cryptography and network protocols. Once Web3Auth provides the user's private key, MeshSDK:

- Uses Cardano's hierarchical deterministic (HD) wallet standards (CIP-1852), which extend BIP32 for Ed25519 keys, to derive the user's wallet addresses.

- Generates **used addresses** (addresses with transaction history) and **unused addresses** for receiving funds and interacting with smart contracts.

- Provides a wallet abstraction that can sign transactions and interact with Cardano nodes.

This integration allows the dApp to treat Web3Auth wallets as native Cardano wallets seamlessly.

3.3 Security Considerations

- **Private Key Safety:** Private keys never leave the user's device and are never stored on servers unencrypted.

- **Key Derivation Path:** Following CIP-1852 standards ensures compatibility with other Cardano wallets and tools.

- **Session Management:** Web3Auth manages secure sessions, and the dApp should handle session expiration and re-authentication carefully.

- **Backup and Recovery:** While Web3Auth provides social login backup, users should be informed about recovery options and limitations.

3.4 Sample Workflow

1. User opens the dApp and selects a social login option.
2. Web3Auth authenticates the user and generates the private key.
3. MeshSDK receives the private key and derives the Cardano wallet addresses.
4. The dApp displays the wallet address and allows the user to sign transactions, such as sending encrypted chat messages or submitting Merkle roots.

4. Encrypted Chat Data Storage on Blockchain

Ensuring the confidentiality and integrity of user chat data is fundamental in a decentralized messaging system. This section discusses how to encrypt chat messages, manage encryption keys, and store encrypted data efficiently on the Cardano blockchain.

4.1 On-Chain vs. Off-Chain Storage

Storing all chat data directly on-chain is impractical due to high transaction costs, limited storage capacity, and performance constraints. Instead, a hybrid approach is adopted:

- **Off-Chain Storage:** Encrypted chat messages are stored off-chain in distributed storage systems (e.g., IPFS, Arweave, or centralized servers) to handle large data volumes efficiently.
- **On-Chain Anchoring:** Cryptographic commitments (hashes) or Merkle roots representing batches of encrypted messages are stored on-chain as transaction metadata. This provides immutable proof of message integrity and ordering.

4.2 Encryption Schemes for Chat Messages

Symmetric Encryption

- Messages are encrypted using a symmetric encryption algorithm such as AES-256-GCM, which provides confidentiality and data integrity via authenticated encryption.

- Symmetric keys are shared securely between communicating parties using asymmetric encryption or key agreement protocols.

Asymmetric Encryption and Key Exchange

- Users generate asymmetric key pairs (e.g., Ed25519 or Curve25519) compatible with Cardano's cryptography.

- Public keys are exchanged or published to enable secure symmetric key distribution.

- Protocols such as Diffie-Hellman key exchange or elliptic curve integrated encryption scheme (ECIES) facilitate secure key sharing.

4.3 User Key Management

- Each user's private keys are securely managed via Web3Auth, abstracting complexities and enabling secure signing and decryption operations.

- Symmetric session keys used for encrypting chat messages are derived per conversation or per message to ensure forward secrecy.

- Key rotation policies and secure backup mechanisms must be considered to prevent key compromise.

4.4 Encrypted Message Format

Encrypted chat messages typically include:

- Initialization Vector (IV) for AES-GCM encryption

- Ciphertext

- Authentication Tag (for integrity verification)

- Optional metadata (timestamp, sender ID) encrypted or hashed as needed

The message is serialized and stored off-chain, with its hash included in the Merkle tree for on-chain anchoring.

4.5 Transaction Metadata for On-Chain Anchoring

- Cardano allows embedding arbitrary metadata (up to approximately 16 KB) in transactions.
- Merkle roots computed from batches of encrypted message hashes are stored as metadata keys/values.
- This mechanism provides tamper-proof evidence of message existence and ordering without revealing content.

4.6 Data Retrieval and Decryption Workflow

- When retrieving chat history, clients fetch encrypted messages from off-chain storage using references proven by on-chain Merkle roots.
- Users decrypt messages locally using their keys.
- Integrity is verified by checking the Merkle proofs against the on-chain root.

5. Merkle Tree Construction for Chat Messages

Ensuring data integrity and efficient verification in a decentralized chat system requires a robust cryptographic mechanism. Merkle trees provide an effective solution by allowing compact proofs of message inclusion and tamper-evidence for large datasets. This section describes the design and utilization of Merkle trees for encrypted chat messages on the Cardano blockchain.

5.1 Purpose of Merkle Trees in Chat Data

- **Data Integrity:** Each chat message is cryptographically hashed and incorporated into a Merkle tree, creating a single root hash that uniquely represents the entire message set.

Any alteration in the messages changes the root, revealing tampering.

- **Efficient Verification:** Merkle proofs enable users or smart contracts to verify the inclusion of specific messages without needing to process the entire dataset, enhancing scalability and performance.

- **On-Chain Anchoring:** The Merkle root is stored as metadata on-chain, anchoring off-chain stored encrypted messages to an immutable ledger and providing a trustless verification point.

5.2 Merkle Tree Structure

- **Leaves:** Each leaf node represents the cryptographic hash of an encrypted chat message (including its IV, ciphertext, and authentication tag).

- **Internal Nodes:** Each internal node is the hash of the concatenation of its two child nodes.

- **Root Node:** The Merkle root, derived from recursively hashing pairs of child nodes up to the top, serves as a compact commitment to the entire message set.

5.3 Building the Merkle Tree

1. **Hash Encrypted Messages:** Apply a secure hash function (e.g., SHA-256) to each encrypted message to create leaf nodes.

2. **Pairwise Hashing:** Recursively hash pairs of adjacent nodes to build higher tree levels. If there is an odd number of nodes, duplicate the last node to complete the pair.

3. **Compute Root:** Continue until a single root hash is produced.

5.4 Merkle Proofs for Message Verification

- A **Merkle proof** is a sequence of hashes required to recompute the Merkle root from a leaf node.

- To prove the inclusion of a message, a user provides the encrypted message's hash and the Merkle proof path.

- Verifiers hash the provided leaf and iteratively combine it with sibling hashes in the proof to recompute the root. If the recomputed root matches the on-chain root, the message's integrity and inclusion are confirmed.

5.5 Storing Merkle Roots On-Chain

- Merkle roots are included in Cardano transactions as metadata fields, typically keyed for easy retrieval.

- This on-chain anchoring provides immutable, publicly verifiable proof of message integrity without exposing message contents.

5.6 Benefits for Decentralized Chat

- **Scalability:** Only small proofs are needed to verify messages rather than the entire message database.

- **Privacy:** Only hashes and proofs are stored on-chain, keeping messages encrypted and private.

- **Auditability:** Third parties can verify message authenticity and ordering without accessing plaintext.

6. Privacy-Preserving Search Using Zero-Knowledge Proofs

Searching encrypted chat data stored on a public blockchain poses a significant privacy challenge. Revealing keywords or plaintext messages during search compromises user confidentiality. Zero-Knowledge Proofs (ZKPs) offer a solution by enabling users to prove that a keyword exists within encrypted data without revealing the keyword or the messages themselves. This section explores the design and implementation of privacy-preserving search mechanisms leveraging ZKPs.

6.1 Challenges of Searching Encrypted Data

- **Data Confidentiality:** Direct search requires decrypting messages, exposing sensitive information.

- **Public Ledger Transparency:** Storing decrypted or plaintext search queries on-chain undermines privacy.

- **Computation Complexity:** Performing secure search on encrypted data is computationally intensive.

6.2 ZKP-Based Keyword Search Overview

- Users generate a zero-knowledge proof demonstrating the existence of a specific keyword within one or more encrypted messages.

- The proof confirms keyword presence and message integrity (via Merkle tree inclusion) without revealing any message content or the keyword itself.

- Verification can be performed on-chain or off-chain, ensuring trustless validation.

6.3 Designing the ZKP Circuit

- **Inputs:**

    - Private inputs: encrypted message ciphertexts, keyword, Merkle proof path.

    - Public inputs: Merkle root stored on-chain.

- **Circuit Tasks:**

    - Verify that a decrypted message contains the keyword (or that the committed hash matches).

    - Validate the Merkle proof linking the message to the on-chain Merkle root.

    - Produce a proof that satisfies both conditions without leaking data.

6.4 Integration with Merkle Trees

- The encrypted message's hash is a leaf node in the Merkle tree.
- The ZKP circuit verifies that this leaf is part of the tree corresponding to the on-chain root.
- This linkage ensures that the search proof is anchored to the immutable blockchain state.

6.5 Performance and Scalability Considerations

- ZKP generation can be computationally expensive; thus, it is often performed off-chain.
- Verification is optimized for speed and can be done on-chain within smart contracts or by trusted off-chain verifiers.
- Techniques like batching proofs or succinct proof systems (e.g., zk-SNARKs) enhance scalability.

6.6 Practical Approaches and Optimizations

- Precompute keyword hashes to avoid expensive plaintext substring search inside circuits.
- Use searchable encryption or Oblivious RAM (ORAM) techniques to reduce circuit complexity.
- Limit search scope to recent message batches or conversation threads to optimize proof size.

6.7 User Experience Flow

1. User submits a keyword search query via the client.
2. Client generates a ZKP proving the keyword exists in messages committed by the on-chain Merkle root.
3. Proof is submitted for verification without exposing messages or keywords.

4. Upon successful verification, user retrieves matching encrypted messages off-chain and decrypts locally.

7. System Architecture and Workflow

This section describes the overall architecture of the decentralized encrypted chat system on Cardano and details the interactions among its components, providing a seamless and secure user experience.

7.1 High-Level Architecture Components

Component	Description
Client Application	User interface for chatting, encryption/decryption, and proof generation.
Web3Auth Authentication	Manages user identity and private key generation via social logins.
MeshSDK Wallet Integration	Derives Cardano wallet addresses and signs transactions securely.
Off-Chain Storage	Stores encrypted chat messages (e.g., IPFS, Arweave, or centralized servers).
Cardano Blockchain	Stores Merkle roots of message batches as metadata, ensuring immutability and verification.
Zero-Knowledge Proof System	Enables privacy-preserving keyword search and message integrity verification.

7.2 End-to-End Workflow

7.2.1 User Authentication and Wallet Setup

- The user logs into the dApp using Web3Auth with their preferred social login.

- Web3Auth generates and manages the user's private key securely on the client side.
- MeshSDK derives the user's Cardano wallet address from this private key for blockchain interactions.

7.2.2 Sending Encrypted Chat Messages

- The client encrypts the message locally using symmetric encryption with session keys shared between chat participants.
- The encrypted message, along with associated metadata (IV, authentication tag), is sent to off-chain storage.
- The hash of the encrypted message is computed and added as a leaf in the local Merkle tree maintained by the client.
- Periodically, the client or a backend service submits the Merkle root of the current message batch to the Cardano blockchain as transaction metadata, anchoring message integrity.

7.2.3 Verifying Message Integrity

- When users fetch chat history, they retrieve encrypted messages from off-chain storage.
- The client obtains the on-chain Merkle root and corresponding Merkle proofs for each message.
- Using Merkle proofs, the client verifies that each encrypted message is part of the anchored Merkle root, ensuring data integrity and authenticity.

7.2.4 Privacy-Preserving Keyword Search

- Users generate zero-knowledge proofs off-chain demonstrating that a keyword exists in the encrypted messages associated with the on-chain Merkle root, without revealing the keyword or message contents.
- The proof is submitted to the blockchain or a verification service, which validates it against the stored Merkle root.

- Upon successful verification, users retrieve matching encrypted messages and decrypt them locally.

7.3 Communication Protocols and Security

- All communication between clients and off-chain storage is secured via encrypted channels (e.g., HTTPS/TLS).

- Key exchanges and session key management use secure asymmetric cryptographic protocols.

- Transactions submitted on-chain are signed via MeshSDK using user-managed private keys, ensuring authenticity.

7.4 Summary

User Device
- Web3Auth: Authentication & Key Management
- MeshSDK: Wallet Derivation & Transaction Signing
- Encrypt Chat Message -> Off-chain Storage (IPFS/Arweave)
- Update Merkle Tree & Submit Merkle Root -> Cardano Blockchain
- Generate ZKP for keyword search -> Submit Proof -> Verification
- Retrieve & Decrypt Messages Locally

8. Security and Privacy Analysis

A core objective of the proposed decentralized chat system is to ensure robust security and strong privacy guarantees for its users. This section analyzes potential attack vectors, threat mitigations, and the privacy-preserving features inherent to the system design.

8.1 Threat Model

- **Adversary Types:**

    - External attackers attempting to intercept or tamper with messages.

    - Malicious insiders or compromised nodes trying to access private chat data.
    - Blockchain observers seeking to infer message content or user behavior from on-chain data.
- **Assets to Protect:**
    - Confidentiality of chat messages.
    - Integrity and authenticity of message data.
    - User private keys and authentication credentials.
    - Privacy of user search queries and communication patterns.

8.2 Confidentiality

- All chat messages are encrypted end-to-end using AES-256-GCM or equivalent strong symmetric encryption, preventing eavesdropping even if messages are stored off-chain or intercepted.
- Encryption keys are exchanged securely using elliptic curve cryptography (e.g., Ed25519/Curve25519), ensuring only intended recipients can decrypt messages.
- Web3Auth protects private keys through distributed key management, minimizing risks of key leakage.

8.3 Integrity and Authenticity

- Merkle trees anchor message batches on the Cardano blockchain, providing tamper-evident proofs of message integrity.
- Merkle proofs enable verification of message inclusion without revealing plaintext, ensuring authenticity.
- Transactions and message metadata are signed with user keys managed via MeshSDK, preventing unauthorized modifications.

8.4 Privacy Preservation

- Encrypted messages and Merkle roots reveal no plaintext information on-chain or off-chain.
- Zero-Knowledge Proofs enable private keyword search without exposing keywords or message contents, preventing inference attacks on search behavior.
- On-chain metadata stores only cryptographic commitments, minimizing data leakage.

8.5 Resistance to Common Attacks

Attack Vector	Mitigation
Man-in-the-Middle (MITM)	Use of TLS for data transmission; end-to-end encryption of messages
Replay Attacks	Use of message timestamps, nonces, and on-chain anchoring with unique Merkle roots
Key Compromise	Web3Auth's distributed key storage and session management; key rotation protocols
Data Tampering	Merkle tree proofs and blockchain immutability prevent undetected modification
Metadata Analysis	Minimal metadata on-chain; encrypted communications; ZKP for search reduces leakage

8.6 Limitations and Future Work

- While ZKP-based keyword search preserves privacy, generating and verifying proofs may introduce latency and computational overhead.
- The off-chain storage layer must be secured and decentralized to prevent censorship and data loss.
- Further enhancements may include anonymous communication channels, mixnets, or decentralized identity integration.

9. Zero-Knowledge Proofs for Private Keyword Search

9.1 Motivation

In a blockchain-based chat system where all messages are encrypted and stored immutably, users may still want to search for specific keywords without revealing their messages or the keywords themselves to other parties. This requirement arises in scenarios like:

- **Compliance checks** without leaking unrelated chat data
- **Personal archival retrieval** of messages containing certain terms
- **Private investigations** where only proof of keyword presence is needed

Zero-Knowledge Proofs (ZKPs) provide a mechanism for this, allowing a user to prove the existence (or non-existence) of a keyword in their encrypted data **without exposing the plaintext**.

9.2 Cryptographic Approach

We adopt a **ZKP-friendly searchable encryption** mechanism, combined with **Merkle proofs**, to verify search results:

1. **Encrypted Messages**
    - All chat messages are encrypted using symmetric encryption (AES-256-GCM).
    - Encryption keys are derived from user private keys stored in Web3Auth MPC wallet.
2. **Searchable Index Creation**
    - Each message generates a **hashed keyword index** using SHA-256 or Poseidon hash (ZKP-friendly).
    - Example: Keyword “Cardano” → hash(keyword, salt) → stored in off-chain index or as part of on-chain Merkle tree.
3. **Merkle Tree Commitment**
    - All hashed keyword entries are committed into a Merkle root stored on-chain.

- This ensures integrity and immutability of the searchable index.

4. **Zero-Knowledge Search Proof**
    - When searching for a keyword, the client:

      a. Computes the keyword hash locally.

      b. Generates a ZKP that the keyword exists in the committed Merkle tree without revealing the keyword.

    - Protocols like **zk-SNARKs** or **zk-STARKs** are used.

    -

9.3 Example Search Flow

Step 1: User wants to search for "staking" in their encrypted chat history.

Step 2: User locally hashes "staking" and uses the salt stored locally.

Step 3: Using ZKP, user proves that this hashed keyword exists in the Merkle tree committed on-chain.

Step 4: The verifier (e.g., the chat app backend or another user) checks the proof against the on-chain Merkle root.

Step 5: If valid, the verifier is convinced "staking" exists in the dataset, but learns nothing else.

9.4 Benefits

- **Privacy Preserving:** No keyword or plaintext messages are leaked.
- **Tamper-Proof:** Merkle root guarantees index integrity.
- **Scalable:** Search proofs are small in size and verifiable in milliseconds.
- **Decentralized:** No need for trusted third parties to perform search.

9.5 Limitations

- **Index Update Cost:** Updating Merkle tree on-chain for each new message adds gas cost.
- **Preprocessing Overhead:** Generating and committing indexes adds extra computation.
- **Search Latency:** ZKP generation time can be noticeable for large datasets, but can be mitigated with pre-computation.

10. Privacy and Security Considerations

The architecture outlined in this white paper is designed to ensure confidentiality, integrity, and verifiability of user communications while operating within the constraints of the Cardano blockchain. Given the immutable and transparent nature of blockchain technology, special measures must be implemented to protect sensitive user data.

10.1 Data Confidentiality

- **End-to-End Encryption (E2EE):**

  All chat messages are encrypted on the client side before transmission. Only participants in the conversation hold the decryption keys.

- **Merkle Tree Data Organization:**

  Message hashes are stored in a Merkle tree structure, allowing verification of message integrity without revealing plaintext content.

- **Off-Chain Encrypted Storage:**

  Encrypted messages are stored off-chain (e.g., IPFS or Arweave) with only message hashes or Merkle roots committed to the Cardano blockchain.

10.2 Zero-Knowledge Proofs (ZKP) for Private Search

- **Privacy-Preserving Keyword Search:**

  A Zero-Knowledge Proof protocol allows a user to prove that a keyword exists in their encrypted chat history without revealing the content of other messages.

- **No Index Leakage:**

  The ZKP mechanism is designed to prevent leakage of message indices or frequency of

search terms.

10.3 Data Integrity and Authenticity

- **Merkle Root Commitment:**

  The Merkle root of the chat history is periodically committed to the blockchain, ensuring tamper-proof verification.

- **Signature Verification:**

  Messages include digital signatures tied to the sender's wallet address, verifiable against their Web3Auth-derived Cardano key pair.

10.4 Key Management and Recovery

- **MPC Wallet Integration:**

  Private keys are never fully exposed; Web3Auth and MPC-based custody ensure secure key handling.

- **Key Recovery Mechanisms:**

  Social recovery or multi-factor authentication methods can be implemented to mitigate the risk of losing access to chat data.

10.5 Compliance and Regulatory Considerations

- **GDPR and Data Deletion:**

  While blockchain data is immutable, encryption ensures that once encryption keys are destroyed, the data becomes effectively inaccessible.

- **Jurisdictional Considerations:**

  Deployment must account for cross-border data privacy laws, especially in regions with strict communication surveillance regulations.

11. Conclusion and Future Work

In this white paper, we presented a novel architecture for integrating **Web3Auth**, **MeshSDK**, **Merkle Trees**, and **Zero-Knowledge Proofs (ZKPs)** into a **Cardano-based decentralized chat system**. By leveraging **Web3Auth** for secure wallet onboarding, **MeshSDK** for seamless Cardano wallet address derivation, and **Merkle Trees** for message integrity verification, our system ensures that all chat data is encrypted, verifiable, and immutable on-chain. Furthermore, by integrating **ZKPs**, we enable keyword search within encrypted chat data without compromising user privacy.

This architecture addresses several key challenges in decentralized communications:

- **Privacy preservation** – Messages remain encrypted end-to-end while still being searchable via ZKP-based queries.
- **Data integrity** – Merkle proofs allow participants to verify that chat data has not been tampered with.
- **User-friendly onboarding** – Web3Auth enables easy access for non-technical users without sacrificing security.
- **Blockchain immutability** – Cardano ensures that chat records are permanent and auditable.

However, there are still open challenges and opportunities for enhancement:

Future Work:

1. **Performance Optimization** – Further work is needed to optimize Merkle Tree generation and ZKP proving times for large chat histories.
2. **Cross-Chain Interoperability** – Exploring the integration of other blockchains for multi-chain chat interoperability.
3. **Scalable Storage Solutions** – Investigating hybrid storage models combining on-chain proofs with off-chain encrypted storage for improved scalability.
4. **Advanced Privacy Mechanisms** – Incorporating Fully Homomorphic Encryption (FHE) to allow more complex queries over encrypted data.
5. **Mobile and Edge Integration** – Developing lightweight mobile and IoT-friendly implementations of ZKP and Merkle proof verification.

The proposed system is a step toward a **fully decentralized, privacy-preserving communication protocol** that can be adopted not only for chat applications but also for secure

messaging in enterprise, government, and financial sectors. With continuous research and community collaboration, we envision a future where **privacy, security, and decentralization are the default standard for digital communications**.

References

1. Hoskinson, C., & Wood, G. *Cardano Documentation*. IOHK. https://docs.cardano.org/
2. Mesh SDK. *Cardano Mesh SDK Documentation*. https://meshjs.dev/
3. Web3Auth. *Web3Auth Documentation*. https://web3auth.io/docs/
4. Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf
5. Merkle, R. C. (1988). *A Digital Signature Based on a Conventional Encryption Function*. Advances in Cryptology — CRYPTO ’87.
6. Ben-Sasson, E., Chiesa, A., Tromer, E., & Virza, M. (2014). *Scalable Zero Knowledge via Cycles of Elliptic Curves*. Theory of Cryptography Conference (TCC).
7. Boneh, D., & Shoup, V. (2020). *A Graduate Course in Applied Cryptography*. Stanford University.
8. Arweave. *Arweave Documentation*. https://docs.arweave.org/
9. Cardano Foundation. *Plutus Smart Contracts*. https://plutus.cardano.foundation/
10. Web3Auth. *Secure MPC Key Management for Web Applications*. https://web3auth.io/docs/mpc/