HASH: Decentralized Messaging Application Using Smart Contracts

Bharathi Koduri [1], Giridhar Mesram [2], Haricharan Vallabhaneni [3], Dr. D.Vijaya Lakshmi
[1,2,3]*Student, Mahatma Gandhi Institute of Technology*
[4]*Head of the Dept, Department of IT, Mahatma Gandhi Institute of Technology*

Abstract — Conventional text messaging systems face limitations due to centralized architectures, compromising network reliability, availability, and privacy. This project proposes a decentralized peer-to-peer solution built on Ethereum to address these shortcomings. The transition to a decentralized system leverages Ethereum's capabilities for secure communication channels, ensuring heightened privacy and reliability. A key innovation is the implementation of a provisional node management system, enabling seamless message transmission even when the recipient is offline. This system optimizes unsent message queue size, preventing excessive resource consumption while maintaining efficient delivery.

The proposed decentralized peer-to-peer model, fortified by Ethereum integration and the resilient node management system, undergoes rigorous testing, confirming its ability to overcome the limitations of traditional centralized approaches and marking a significant advancement towards a more reliable, private, and decentralized messaging infrastructure.

Index Terms— decentralized, peer-to-peer solution, Ethereum, reliability, Smart Contracts

I. INTRODUCTION

In the realm of emerging technologies, Hash emerges as a beacon of promise within the blockchain landscape. Beyond its origins in cryptocurrency, blockchain technology is fundamentally reshaping how we handle data—revolutionizing storage, updates, and transfers cross networks. As the world becomes increasingly interconnected, the decentralized power inherent in blockchain takes center stage. Utilizing consensus mechanisms and voluntary adherence to social contracts, blockchain facilitates the creation of a decentralized value-transfer system— a system that is globally accessible and virtually free to use. Ethereum, a trailblazing project, aims to provide a versatile technology foundation supporting various transaction-based state machine concepts. Moreover, Ethereum aspires to furnish end-developers with an integrated end-to-end system, introducing a trustful object messaging compute framework within a previously unexplored computing paradigm. Enter Hash—a decentralized messenger redefining the boundaries of conventional messaging applications. Hash empowers users to engage in chat, seamless transactions, and even earn rewards—all without the limitations of censorship or privacy concerns. By facilitating direct connections between users without relying on third-party intermediaries, Hash ensures a secure and private messaging experience. Users can send and receive messages within the decentralized application (DApp), enabling direct communication free from centralized oversight. This project represents a significant stride towards unlocking the full potential of blockchain technology, transforming the landscape of digital communication and transactions.

Blockchain has the potential and capacity to improve online security and user's trust. Global information transmission has become increasingly accessible. Through the power of the default, consensus mechanisms and voluntary respect of the social contract that it is possible to use the internet to make a decentralized value-transfer system, shared across the world and virtually free to use. Ethereum is a project, which attempts to build the generalized technology, a technology on which all transaction-based state machine concepts may be built. Moreover, it aims to provide to the end-developer a tightly integrated end-to-end system for building software on a hitherto unexplored compute paradigm in the mainstream: a trustful object messaging compute framework.

Salient Features

- Peer-to-Peer Communication: Decentralized messaging apps facilitate direct communication

between users without relying on central servers. Messages are sent directly between devices, enhancing privacy and reducing the risk of censorship or surveillance.

- End-to-End Encryption: Messages are typically encrypted from sender to recipient, ensuring that only the intended parties can read the content. This encryption prevents intermediaries, including the messaging service provider, from accessing message contents.
- User Privacy: Decentralized messaging apps prioritize user privacy by minimizing the collection of personal data and metadata. Since there's no central authority storing user information, users have more control over their data and are less susceptible to data breaches or privacy violations.
- Censorship Resistance: Because decentralized messaging apps do not rely on central servers, they are more resistant to censorship attempts by governments or other authorities. Users can communicate freely without concerns about messages being blocked or monitored.
- Open Source: Many decentralized messaging apps are built on open-source protocols, allowing anyone to review the code for security vulnerabilities and contribute to its development. This transparency fosters trust among users and promotes innovation within the community.
- Blockchain Integration: Some decentralized messaging apps leverage blockchain technology to enhance security, facilitate peer discovery, or incentivize network participation. Blockchain can provide tamper-proof message histories and enable features like decentralized identity verification**.**

Objective

To develop a decentralized, peer-to-peer messaging system built on the Ethereum blockchain technology, addressing the limitations of conventional centralized messaging systems in terms of reliability, availability, and privacy.

Motivation

The primary motivation is to address the inherent drawbacks of centralized messaging architectures by exploiting the advantages of decentralized peer-to-peer communication and the Ethereum blockchain, ultimately creating a more reliable, private, and efficient messaging solution.

Problem Statement

Privacy and reliability are important aspects of information transfer. The messaging applications are one way to communicate between two people. To be confident about sharing information on the application, users must trust the medium that is sending the messages. A centralized server is likely to be shutdown at some point during the updating or adding new features or even when a single person or organization has authority over the server. To avoid this, we need a decentralized system that does not have a single point of failure.

Existing System

The current landscape of messaging applications predominantly relies on centralized architectures. Popular platforms like WhatsApp, Messenger, and Telegram utilize centralized servers to facilitate message transmission and storage. This centralized model often poses challenges related to data privacy, security, and potential single points of failure. Users' messages are typically routed through these central servers, raising concerns about privacy and vulnerability to external threats. Users of existing messaging systems are subject to the policies and terms of service dictated by the central authorities operating these platforms. This dependency on third-party intermediaries not only raises concerns about data ownership and security but also introduces the risk of censorship and surveillance. In the current paradigm, users have limited control over their data. Messages are stored on centralized servers, leaving users vulnerable to data breaches or unauthorized access. Moreover, the lack of end-to-end encryption in some instances compromises the confidentiality of conversations.

Limitations

1. Privacy concerns: User messages and data are accessible to the central authorities operating the platforms, compromising privacy.
2. Security vulnerabilities: Centralized systems present a single point of failure, making them susceptible to hacking, DDoS attacks, and other security breaches.
3. Lack of data ownership: Users have limited control over their data, which is stored on

centralized servers owned by third-party companies.
4. Limited end-to-end encryption: Some platforms may not provide end-to-end encryption, compromising the confidentiality of user conversations.
5. Dependence on central authorities: Users are bound by the policies and decisions of the central authorities operating these messaging platforms, limiting their control and autonomy.

Proposed System

Hash proposes a paradigm shift by adopting a decentralized architecture built on blockchain technology. The system leverages smart contracts to facilitate secure, direct communication between users without relying on centralized servers. In the proposed system, Hash prioritizes user privacy by eliminating the need for messages to pass through central servers. The use of smart contracts ensures end-to-end encryption, enhancing the confidentiality and security of user conversations. Users have greater control over their data, reducing the risk of unauthorized access. Hash empowers users with censorship-resistant communication. By removing third-party intermediaries, the system ensures that users have direct control over their messages, free from external interference or surveillance.

The adoption of blockchain technology, specifically Ethereum, provides additional advantages such as transparency, immutability, and resilience. Smart contracts enable the creation of a trustful object messaging compute framework, contributing to a more robust and reliable messaging ecosystem.

II. ALGORITHMS

SHA-256

SHA-256, or Secure Hash Algorithm 256-bit, is a cryptographic hash function that belongs to the SHA-2 family of hash functions. It is widely used for ensuring data integrity and security. - In the "Hash" messaging application, SHA-256 can be integrated into the message encryption and verification process. When a user sends a message, the application can compute the SHA-256 hash of the message content. This hash, along with other relevant information, can be included in the message payload or used to generate a digital signature.

- Upon receiving the message, the recipient can independently compute the SHA-256 hash of the received message content. If the computed hash matches the transmitted hash, it verifies the integrity of the message. This process ensures that the message has not been tampered with during transmission and maintains the privacy and security of user communication within the decentralized messaging platform.

- Message Integrity -SHA-256 generates a fixed-size (256-bit) hash value from any given input data. This hash is unique to the input, and even a small change in the input will result in a drastically different hash. In the context of messaging, when a user sends a message, the SHA-256 algorithm can be applied to the message content. The generated hash can then be transmitted along with the message. Upon receipt, the recipient can recompute the hash from the received message and compare it to the transmitted hash. If they match, it indicates that the message content has not been altered during transmission.
- Digital Signatures - SHA-256 is commonly used in conjunction with public-key cryptography to create digital signatures. In the messaging application, users can sign their messages using their private key, and the recipient can verify the signature using the sender's public key. The hash of the message is an essential component of the digital signature process. It ensures that the signed content has not been tampered with and that the signature is valid.
- Password Hashing - If your messaging application involves user accounts and passwords, SHA-256 can be used for secure password hashing. Instead of storing actual passwords, the system can store the hash of the password. During login, the system hashes the entered password and compares it to the stored hash. This way, even if the password database is compromised, the actual passwords remain secure.
- Data Integrity in Smart Contracts - In a decentralized application utilizing smart contracts, SHA-256 can be employed to verify the integrity of data stored on the blockchain. Smart contracts can use the hash of certain data as a

reference point to ensure that the stored data has not been manipulated.

Smart Contracts

Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They run on blockchain technology and automatically enforce, verify, or facilitate the execution of contract terms without the need for intermediaries. These digital contracts enable trustless and decentralized transactions, as the rules are embedded in code and executed by the blockchain network.

- Message Encryption and Decryption -- Users initiate a smart contract to encrypt their messages using the recipient's public key. Recipients, upon receiving the encrypted message, use their private keys to interact with a decryption smart contract and retrieve the original message.
- Digital Signatures- When sending a message, users trigger a smart contract to create a digital signature using their private key. Recipients use the sender's public key, stored in a smart contract, to verify the signature and confirm the authenticity of the sender.
- Decentralized Identity Management - Each user has a dedicated smart contract representing their identity on the blockchain. Users can update their identity information through interactions with their identity smart contract.
- Smart Contract Wallets - Users manage their cryptocurrency holdings through smart contract wallets. Interactions with these wallets, facilitated by smart contracts, enable secure transactions within the messaging application.
- Decentralized Storage of Messaging Hashes - Smart contracts store message hashes on the blockchain. Users can access these hashes to verify the integrity of their messages, ensuring that the content remains unchanged over time.

III. SOFTWARE USED

Metamask

It is a browser extension for accessing Decentralized Applications. Metamask injects the Ethereum web3 API into the browser, which can be used by developers and users to interact with the Ethereum Blockchain and provides features such as creating accounts, transferring ether, signing transactions in a DApp, and interacting with smart contracts.

Web3.js

It is a collection of libraries that allows applications to interact with a local or remote Ethereum node using HTTP, IPC, or WebSocket. It provides various functions and tools which developers can use to write code for deploying smart contracts, interacting with smart contracts, getting account details, and initiating transactions.

React

React is an open-source frontend framework maintained by Meta and a community of individual developers. It is a component-based User Interface (UI) framework that is used to develop single page applications using reusable UI components

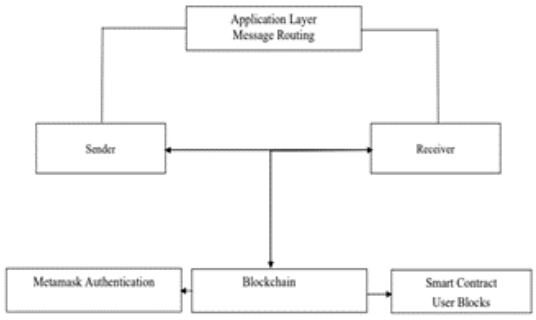

Fig 1. System Architecture

IV. IMPLEMENTATION

This section presents the implementation details of the developed blockchain DApp. Figure 2 shows the implementation set-up used. The frontend of the DApp is a web browser equipped with MetaMask extension. Moreover, Next.js (a React framework) and library functions of Web3.js are used to build the frontend. The backend is a decentralized platform built using Ganache (Ethereum) blockchain and smart contracts written in solidity language. Three different smart contracts have been developed — Main, Post, and Auth contracts.

Main Contract

Main contract consists of various member functions responsible for registering a new user on the platform. It records all the requests for registration and keeps

track of the approved users. The contract also ensures that no two users should have the same address.

Post Contract

Post contract handles the (direct and indirect) posting of messages on the platform. When a user creates and posts a new message, the process is managed by this contract. Furthermore, this contract allows users to re-share an existing post. The post contract keeps track of all the posts (both original and re-shared) along with the users who created those posts.

Authorization Contract

Login and sign-out functionalities (i.e., user authentication) are performed by invoking the Auth contract. The contract reads the data from the deployed Main contract and approves or disapproves a login attempt by a user. If the user is an approved existing user, the login is successful, or the login request is denied.

Various costs of smart contracts developed.

Contract name	Cost (Gwei)	Cost (Ether)	Cost (USD)	Gas Used	Gas Limit
Main Contract	2,396,604	0.002396604	3.953	2,756,095	3,000,000
Post Contract	2,058,742	0.002058742	3.356	2,367,554	3,000,000
Auth Contract	193,981	0.000193981	0.320	223,079	3,000,000

1 Gwei = $1/10^9$ Ether, and 1 Ether = $ 1649.52 on 03.02.2023.

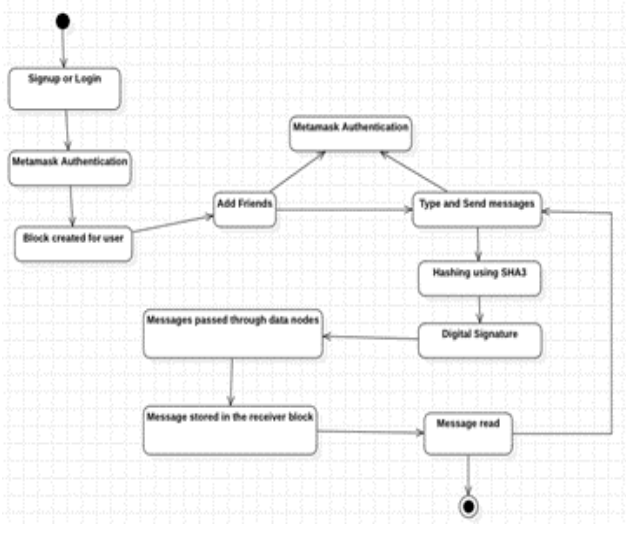

Fig 2. Activity Diagram

Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a System. For our application, decentralized messaging application. A user whenever he wants to signup or login to his account, he enters his credentials and the underlying program goes through metamask authentication for linking his metamask account to the messaging application. A block is created for the user. The user can add friends by typing in the credentials of his friends and this process results in some transaction fees known as gas. Only after any other user is the sender's friend can they send a message to another user. Every action can cost some gas based on the exchange. The sent message goes through hashing process, in this case SHA3 and follows our fundamental principle of passing through the data nodes until the receiver gets the message. The receiver's message is stored in his block

V. RESULTS

Fig 3. Hash Landing Page

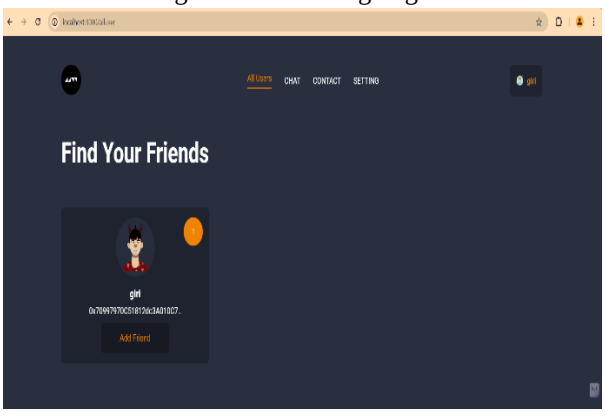

Fig 4. All Users Page

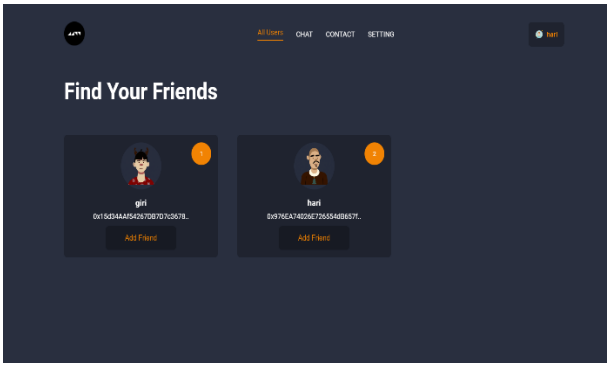

Fig 5. All Users page after adding new friend

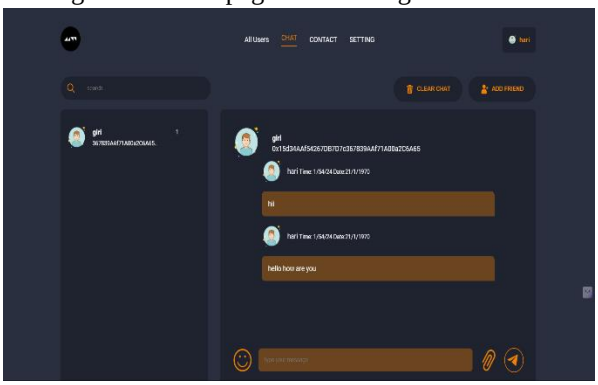

Fig 6. Chat with friend

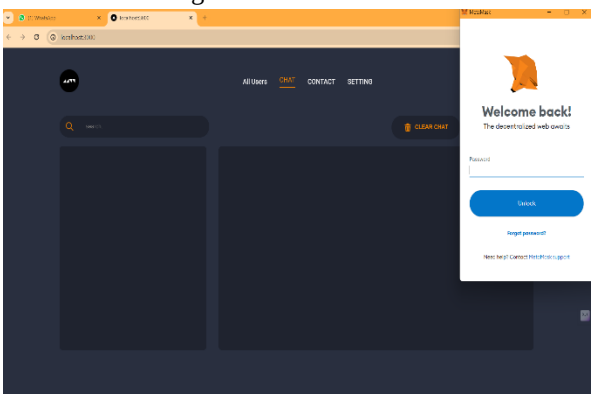

Fig 7. Metamask

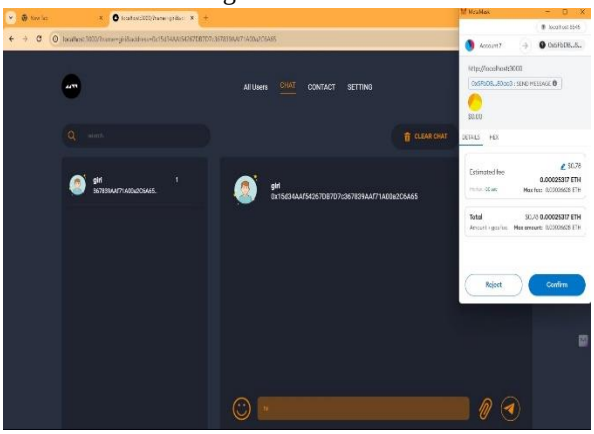

Fig 8. Metamask Wallet with fee

VI. FUTURE SCOPE

The developed ChatApp can be extended to share information globally in the entire education ecosystem, comprising various schools, universities, research institutions, regulatory bodies, and councils. Such a blockchain-based information-sharing fabric can restrict fake or misinformation in an education system which can enable us to overcome challenges involved in the identification of authentic stakeholders (universities, professors, and students), sharing of students' certificates, transparent information exchange during the accreditation process, fair and compliant admission process, and secure management of the intellectual property

VII. CONCLUSION

Web 2.0 applications main base is the centralized server which makes all the messages exchanged stored in a single server. This has two disadvantages, first one being the single point of failure meaning any type of error occurred in the server should be resolved for the application to run properly, the application may be down during the server error. The second disadvantage being the privacy, all our messages can be read by the third person. The peer-to-peer messaging is more secure and reliable than web 2.0 messaging applications. Decentralized environments are a better option than a centralized one since the whole system will shut down in case the central server goes down. The messages sent are secure and protected due to the usage of the technologies like blockchain, metamask, SHA-256 hashing

REFERENCE

[1] Badgujar Niranjan Sanjay, Nair Aditya Sivaram, Nair Sidharth, Vishwakarma Anant, Dr. Sharvari Govilkar "A Decentralized Application for Secure Private and Group Messaging in a Peer-to-Peer Environment", 2022 International Research Journal of Engineering and Technology (IRJET), 1008-1012

[2] Fei Yang, Chunxiao Xing, Yong Zhang, Xiaojun Ye, "Decentralized Distributed Messaging system for Realtime Web Applications and Services" 2014, 11th Web Information System and Application Conference, 2014, IEEE publication (165-171)

[3] U.P Ellewala, W.D.U.H Amarsena, H.V Sachini Lakmali, L.M.K Senanayaka A, N Senarathne, "Secure Messaging Based Platform using Blockchain" 2020 2nd International conference on Advancement on computing ICAC, IEEE Publication (317-322)

[4] Abhishek P. Takale, Chaitanya V. Vaidya, Suresh S. Kolekar " Decentralized Chat Application Using Blockchain Technology" International Journal for Research in Engineering Application & Management (IJREAM) ISSN : 2454-9150 Special Issue - CTRD - 2018

[5] Kahina Khacef, Guy Pujolle, "Secure Peer-to-Peer communication based on Blockchain", 33rd International Conference on Advanced Information Networking and Applications (AINA2019), Matsue, Japan, Mar 2019

[6] Paulo Chainho, Steffen Drusedow, Ricardo Lopes Pereira, Ricardo Chaves, Nuno Santos, Kay Haensge, Anton Roman Portabales, "Decentralized Communications: Trustworthy Interoperability in Peer-To-Peer Networks", 2017.

[7] Naik, Rahul P and Courtois, Nicolas T. 2013. Optimizing the SHA-256 Hashing Algorithm for Faster and More E_cient Bitcoin Mining

[8] Nileshkumar Pandey, Doina Bein, "Web Application for Social Networking using RTC", IEEE Annual Computing and Communication Workshop and Conference (CCWC), 2018.

[9] Peter Menegay, Jason Salyers, Griffin College, "Secure Communications Using Blockchain Technology", Milcom 2018 Track 3 - - Cyber Security and Trusted Computing, 2018.