TagTalk: A Decentralized Social Ecosystem Focused on Privacy and Interests

At **TagTalk**, we prioritize **interests**, **sharing**, and **privacy** in our exploration of a decentralized social ecosystem centered around tags. Our platform is designed to protect users' privacy while allowing them to connect with others who share their interests. Whether you're looking to discover new hobbies or connect with like-minded individuals, **TagTalk** offers a safe and engaging space to explore your passions. Join us on our journey to build a social network that puts users first.

Problem Statement

In the Web 2.0 era, the data is stored on centralized servers, making it inevitable for our communications to be subject to censorship and location tracking. While blockchain technology provides some level of anonymity, it cannot currently meet the performance requirements of online social interactions. Our team aims to combine zero-knowledge technology to create a high-performance instant messaging and social authentication tool that also protects user privacy.

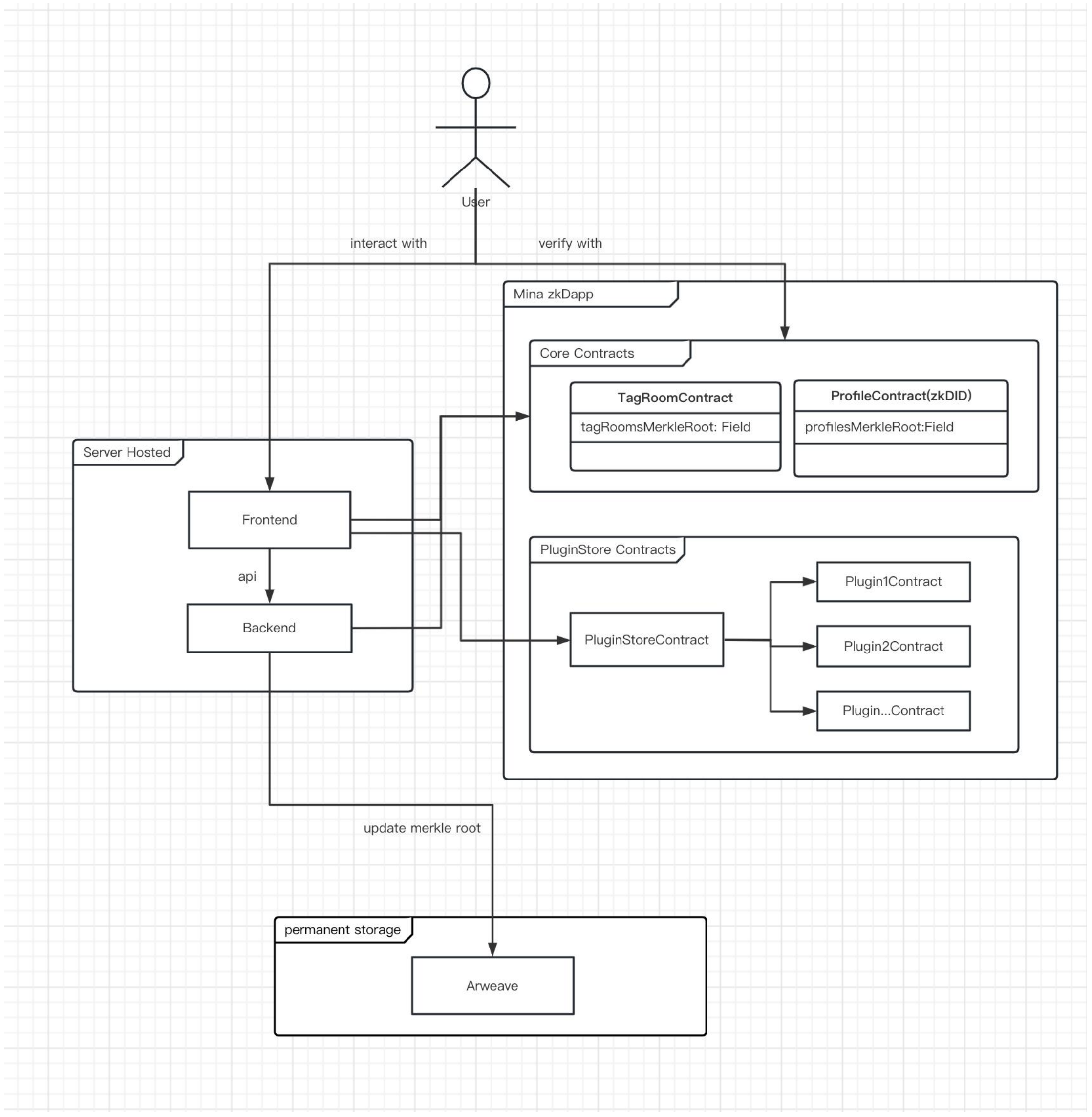
Solution

Through zero-knowledge technology, we can separate off-chain data and on-chain validation. As a result, user social data can be encrypted and stored off-chain, and then validated through a verifiable on-chain Merkle root to ensure data integrity. Meanwhile, off-chain data can also be archived for permanent storage (e.g. on Arweave), thus enhancing data accessibility.

All user chats revolve around Tags. When a Tag is set to public (e.g. interests, hobbies, current events, on-chain transaction id or address), anyone can join the public channel and participate in the conversation. However, when a private chat room is needed, a unique key can be created as the Tag, so that only those who know the key can join the private channel and engage in a private conversation. Private Tags can be the result of a monetary transaction, or secret code known only to a select group of individuals. By allowing both public and private chats with customized Tags, we aim to provide a flexible and customizable chatting experience that can accommodate a wide range of interests and needs.

In addition, our vision of decentralized social media also includes the following features:
Interest-centric: Users engage in conversations centered around tag-based content. Interests can be daily hobbies, topics, or even on-chain addresses and transactions, and are marked by tags. Private chats require users to maintain the uniqueness and privacy of their tags.
Data privacy: Users have complete control over their own data and access rights. Centralized facilities cannot access users' private data or censor their speech.
Composable Lego: By using smart contract plugins based on social behaviors, we can implement more intelligent contract scheduling for zkApp projects and build a widely connected social ecosystem. Some examples of composable Lego scenarios include, but are not limited to: zkDID, conversation proofs, gambling contracts, AA splitting, and payment proof.

Architecture

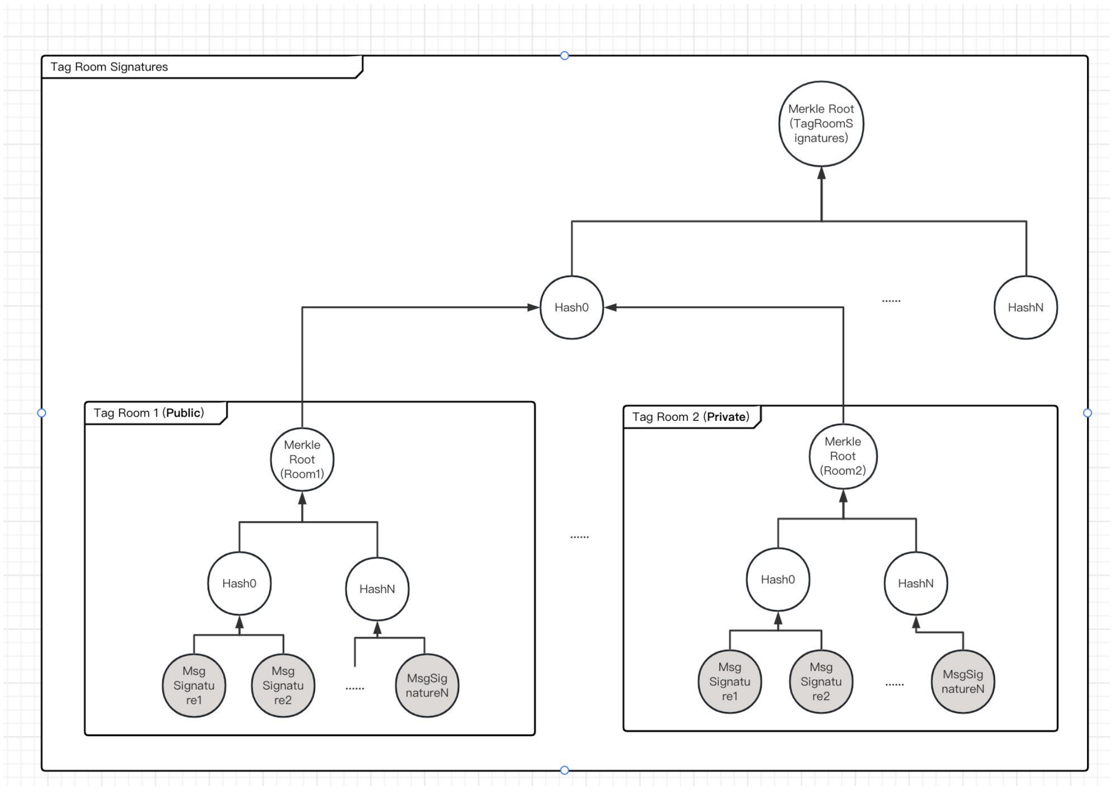


As shown in the attached architecture diagram, the main architecture consists of server front-end and back-end programs, zkApp contracts, and permanent storage archives.

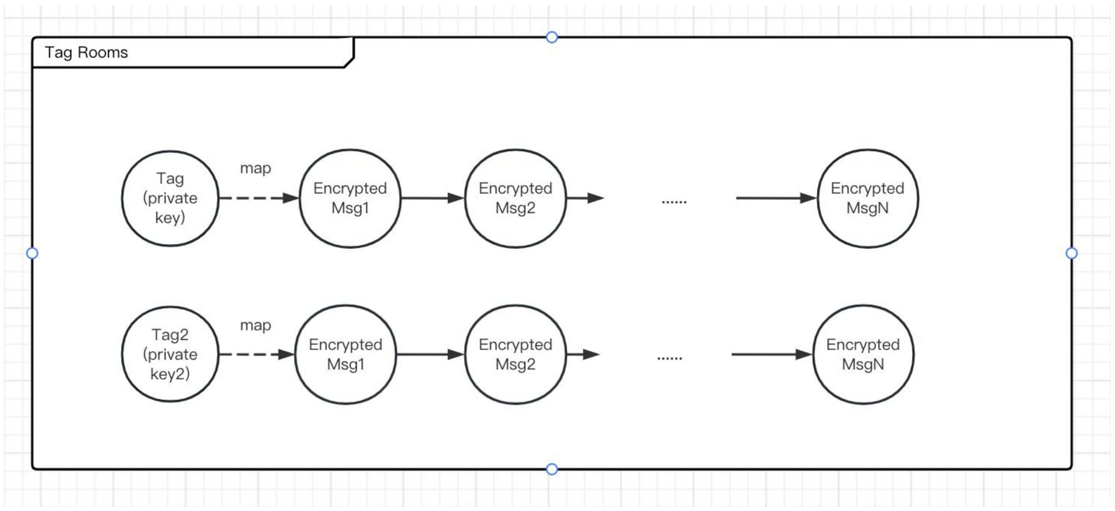
Front-end development: Most of the user interaction experience will be developed here, including connecting wallets, creating chat channels, and viewing and sending chat information through interface interactions.

Server back-end: Provides access API interface for the front-end, stores encrypted chat data for users, and generates a verifiable Merkle root.

Permanent storage archive: The verifiable Merkle root will be stored in the permanent storage archive for backup purposes and to facilitate future verification.



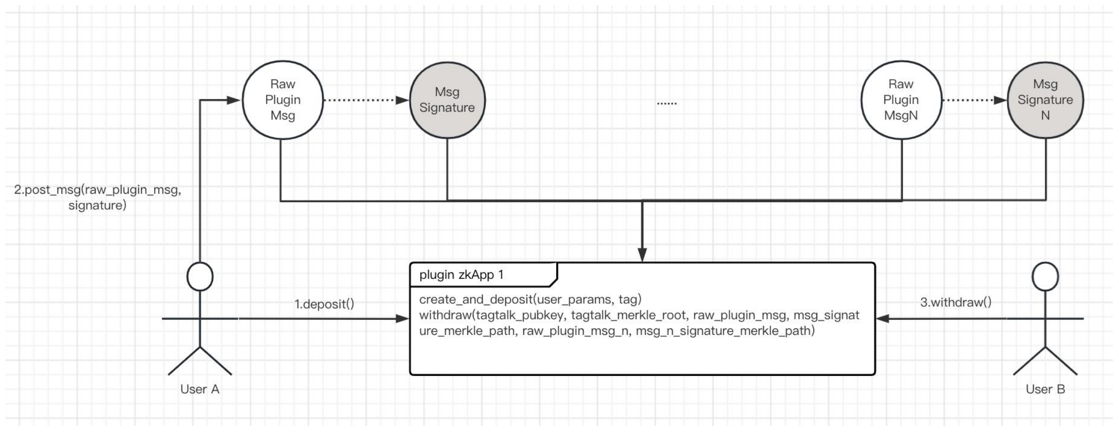
In this system, we center around tags. Tags mark the Tag Room, and when a user enters the Tag Room, all messages in that channel will be encrypted using the Tag as the encryption key. This means that those who know the Tag have both read and write access to the chat group.



As seen in Diagram 2, each Tag Room is represented by a verifiable Merkle tree consisting of a list of messages in that Tag Room. By verifying the Merkle root, we can validate the message list in that discussion group. At the same time, the Merkle roots of all Tag Rooms in the system act as Merkle leaves and can be used to construct a global Merkle tree. The backend can then return the complete Merkle path, which can be used to verify the existence of chat messages.

For a better interactive experience, all Merkle tree data of Tag Rooms in the system is temporarily stored in the backend. Since the data is encrypted, the backend program cannot access the chat data. Additionally, periodic snapshots of the data are taken and stored in permanent storage such as Arweave.

The frontend interacts with Mina's SnarkyJS to validate chat data, while the backend uses the SnarkyJS toolkit to generate Merkle paths and record Merkle roots.



Raw Plugin Msg
Msg Signature
......
Raw Plugin MsgN
Msg Signature N
2.post_msg(raw_plugin_msg, signature)
plugin zkApp 1
create_and_deposit(user_params, tag)
withdraw(tagtalk_pubkey, tagtalk_merkle_root, raw_plugin_msg, msg_signature_merkle_path, raw_plugin_msg_n, msg_n_signature_merkle_path)
1.deposit()
3.withdraw()
User A
User B

As seen in Diagram 3, the backend maintains a hashed tag-to-encrypted message list map, which can be used to index all message lists in a given Tag. The frontend loads these encrypted chat messages and performs local decryption to display the corresponding original chat data in the chat group.

As seen in Diagram 4, we implemented a simplified version of a plugin that allows users to store tokens and stores the plugin data in messages using zkOracle. When the user needs to withdraw tokens, they provide the original plugin data and the Tag Room key and use the zkOracle in conjunction with the system's Merkle root to validate the message's correctness. The plugin generates the corresponding results and returns the stored tokens.

As an example, suppose that the user Alice writes a riddle and sets a reward for the user who guesses it correctly in the same group. Alice then sends tokens to the riddle zkDapp and sets the riddle. The zkDapp generates the corresponding reveal riddle data and stores it in the chat record. When the user Bob guesses the riddle in the chat page, they can use the contract to withdraw the reward that Alice deposited previously by combining the revealed riddle data. At this point, people in the chat group can see the content of the riddle and verify its correctness. Additionally, we can develop other plugins, such as token price oracles, to enable price betting in the chat context. These plugins strike a balance between openness and privacy, making it easier to verify data in the chat context while considering both data privacy and information disclosure.

Existing Work

No

Budget and Milestones

Standard Budget

25000

Standard Scope

We will focus on the core development of social functionality to deliver the MVP product, including user profiles and the implementation of on-chain chat rooms. Plugin functionalities are not included in this budget.

Frontend and backend development of zkApp: $10,000

Core contract development: $10,000

Page design and branding: $3,000

Development support (including server rental, development tool expenses, etc.): $2,000

Standard Scope Milestones

Create an MVP product that can deliver the following features:
Users can use their profile to build their own zkDID.
Chat in public channels based on interests.
Create private channels and chat within them.
The front-end and back-end will be deployed on Vercel, and the code will be open-sourced on GitHub.

Advanced Budget

45000

Advanced Scope

In addition to the standard budget, we will also need to cover the following expenses to support the development costs.
Design and implementation of plugin center contract: $10,000
Code implementation of two example plugins: $10,000

Advanced Scope Milestones

In the advanced budget, we plan to initially implement the core functionality of the plugin center contract and develop one to two plugin instances based on this plugin benchmark. The plugin center requires a well-designed interface to enable other plugin instances to be extended based on this interface.

Risks and Unknowns

This section provides details about the proposal's risks and unknowns. What challenges may be faced, and how might they be dealt with?

Risks

Due to the novelty of Mina technology, we may have limited familiarity with Mina's snarkyJS library, which could hinder our development progress. However, with the strong technical support available, we believe this risk can be overcome.

Proposer Name

Leo Liao

Proposer Github

https://github.com/ilzc

Proposer Experience

Project OnlyBadge : Participated in the web3 Jam competition organized by Flow and made it to the final stage.