Use cases for Chain Signatures

caution This technology is currently inAlpha and should only be used in atestnet environment. Chain signatures enable you to implement multichain and cross-chain workflows in a simple way. Take a look at a few possible use cases:

Trade Blockchain assets without transactions

Trading assets across different blockchains usually require using a bridge that supports them, bringing longer settlement times as the trades are not atomic and require confirmation on both blockchains.

Using Chain signatures you have the ability to change the ownership of different blockchain accounts (e.g., Bitcoin and Ethereum) to trade assets across chains without doing on-chain transactions. This way you can keep native tokens on their native blockchain (e.g.,BTC on Bitcoin,ETH on Ethereum,ARB on Arbitrum), and trade them without bridges. As an added bonus, trades are atomic across chains, settlement takes just 2 seconds, and it supports any token on any chain.

For example, a basic trade flow could be:

1. Users create an account controlled by NEAR chain signatures
2. Users funds these accounts on the native blockchains (depositing)
3. Place orders by funding a new account for the total amount of the order
4. Another user accepts the order
5. Users swap control of the keys to fulfill the order

Details * User A hasETH * on the Ethereum blockchain, and wants to buy native Bitcoin * User B wants to sell Bitcoin for Ethereum

Steps

1. User B, using NEAR, creates and funds a new account on Bitcoin with 1BTC
2. User B, using the spot marketplace smart contract, signs a transaction to create a limit order. This transfers control of the Bitcoin account to the smart contract
3. User A creates a batch transaction with two steps* Creating and funding a new Ethereum account with 10ETH
4.
    - Accepting the order and atomically swapping control of the accounts
5. User A takes ownership of the Bitcoin account with 1BTC
6. , and User B takes ownership of the Ethereum account with 10ETH
7. User A and B can"withdraw"
8. their asset from the order by transferring the assets to their respective"main"
9. accounts

Oauth-controlled Blockchain accounts

On-boarding is a huge problem for decentralized applications. If you want widespread adoption you can't expect people to keep seed phrases safe in order to use an application.

An attractive way of managing Web3 accounts is to use existing Web2 accounts to on-board users. This can be done in the following way:

1. Deploy a NEAR contract that allows the bearer of a user'sJWT token
2. to sign a blockchain transaction (Ethereum, Polygon, Avalanche, and others)
3. The user validates their identity with a third-party receiving a JWT Token
4. The user holding that token can interact with blockchain applications on Ethereum/Polygon/+++ via the NEAR contract for the duration of it's validity

Any method of controlling a NEAR account can also be used to control a cross-chain account.

About JWT tokens JSON Web Tokens are a standard RFC 7519 method for representing claims securely between two parties. They are used in this example to represent the claim that someone is the owner of an Oauth account.

Cross-chain Zero-friction onboarding

Using unique features of the NEAR account modelKeypom provides zero-friction onboarding and transactions on NEAR. They are generally used for NFT drops, FT drops, and ticketing.

A generic Keypom user-flow could be:

1. The developer creates a restricted NEAR account
2. The account is funded withNEAR

3. The user receives a key with limited control of the account
4. The user uses the funded account to call controlled endpoints on NEAR
5. The user returns the remaining funds to the developer and their account is unlocked

tip This allows easy on-boarding to decentralized apps. The accounts are initially restricted to prevent the user being able to simply withdraw theNEAR from the account.

DeFi on Bitcoin (and other non-smart contract chains).

Using chain signatures, smart contracts on NEAR can control externally-owned accounts on non-smart contract chains like Bitcoin, Dogecoin, XRP Ledger, Bittensor, Cosmos Hub, etc. This enables developers to use NEAR as a smart contract “layer” for chains that do not support this functionality natively.

For example, a developer can build a decentralized exchange for Bitcoin Ordinals, using a smart contract on NEAR to manage deposits (into Bitcoin addresses controlled by the contract) and to verify and execute swaps when two users agree to trade BTC for an Ordinal or BRC20 token.

Example:

1. Seller generates a deposit address on Bitcoin that is controlled by the marketplace smart contract on NEAR via chain signatures
2. Seller deposits a Bitcoin Ordinal to the deposit address
3. The Ordinal is listed for sale with a price and a pre-commitment signature from the seller
4. Buyer accepts the order, deposits USDC
5. The control of the Bitcoin Ordinal address is given to the buyer, USDC on NEAR is transferred to the seller

Using Chain Signatures

With Chain Signatures you can do the same but across many chains, for example Polygon:

1. The developer creates a restricted NEAR account with a key
2. The account is funded withNEAR
3. andMATIC
4. The user receives a key with limited control of the account
5. The user uses the funded account to sign payloads calling controlled endpoints on Polygon
6. The user returns the remaining funds to the developer and their account is unlocked

This allows developers to pay for users to use arbitrary contracts on arbitrary chains.

Decentralized Clients

A big problem in decentralized applications is that while the smart contracts are tamper-proof, the clients that access them generally are not. This allows practically complete control over any user account provided they are using the frontend assets that you serve. This has security, trust, and regulatory implications.

When smart contracts can sign payloads you can start usingsigned exchanges (or polyfills) to require HTTP exchanges to be signed by a certain key. If it is not signed with this key the SSL certificate is considered invalid. This means that individual users cannot be served invalid frontends without it being generally observable and non repudiable.

Communication with private NEAR Shards

Companies likeCalimero offer private NEAR shards. Currently, sending messages to and from these NEAR shards is troublesome. If each shard had the ability to sign their message queues, they could be securely sent from one shard to another. Thus you could communicate bidirectionally with any shard as easily as you can with a contract on your own shard.

tip This could also simplify NEAR's sharding model, by treating each NEAR shard like one would a private shardEdit this page Last updatedonMar 22, 2024 byJosh Ford Was this page helpful? Yes No

Previous What are Chain Signatures?Next Data on the Blockchain