I propose an improvement to how Confidential Compute Records (CCRs) are signed in suave-geth

by adopting the EIP-712 signature standard. This change would help front-end developers support CCRs more easily, as they will no longer need to use the deprecated eth_sign

method, which gives scary warning signs when used and will soon be removed entirely by popular wallets like MetaMask.

As this modifies our core transaction type, it is a breaking change and will not be rolled out until a new testnet to preserve the functionality of current SUAPPs.

## Key Changes:

1. EIP-712 Signature Support

: * We've added an Envelope

boolean field to the ConfidentialComputeRecord

structure. This field indicates whether a CCR is signed using the EIP-712 standard.

- Implemented a method EIP712Hash()

to generate the EIP-712 compliant hash for ConfidentialComputeRecord

.

1. We've added an Envelope

boolean field to the ConfidentialComputeRecord

structure. This field indicates whether a CCR is signed using the EIP-712 standard.

1. Implemented a method EIP712Hash()

to generate the EIP-712 compliant hash for ConfidentialComputeRecord

.

1. New EIP-712 Envelope Structure

: * Defined the EIP-712 envelope structure in core/types/suave_eip712.go

. This structure is essential for creating the EIP-712 hash and ensuring compatibility with the standard.

1. Defined the EIP-712 envelope structure in core/types/suave_eip712.go

. This structure is essential for creating the EIP-712 hash and ensuring compatibility with the standard.

1. Enhanced Transaction Handling

: * Updated the transaction marshalling and unmarshalling processes to include the new Envelope

field.

- Modified the transaction signing logic to compute the EIP-712 hash when the Envelope

field is set to true.

1. Updated the transaction marshalling and unmarshalling processes to include the new Envelope

field.

1. Modified the transaction signing logic to compute the EIP-712 hash when the Envelope

field is set to true.

1. Testing and Validation

: * Added comprehensive unit tests to ensure the correct implementation of EIP-712 hash generation and signature validation.

1. Added comprehensive unit tests to ensure the correct implementation of EIP-712 hash generation and signature validation.

2. SDK and CStore Engine Adjustments

: * Introduced a useEIP712

boolean field in the SDK Client

struct to enable or disable EIP-712 signature usage.

- Updated the Confidential Store Engine to handle EIP-712 signed transactions appropriately, ensuring backward compatibility and smooth transition.
- Introduced a useEIP712

boolean field in the SDK Client

struct to enable or disable EIP-712 signature usage.

1. Updated the Confidential Store Engine to handle EIP-712 signed transactions appropriately, ensuring backward compatibility and smooth transition.

Usage:

Here's how you can get started using EIP-712 signatures for CCRs. First, check out the branchfeature/eip712-ccrs locally and start the client as normal, then:

1. SDK Client Configuration

:

client := sdk.NewClient(rpcNode, privateKey, kettleAddress).WithEIP712()

1. Creating and Signing a CCR

:

record := types.ConfidentialComputeRecord{ KettleAddress: kettleAddress, Nonce: nonce, To: &contractAddress, Value: nil, GasPrice: gasPrice, Gas: gasLimit, Data: calldata, ChainID: chainID, Envelope: true, } signedTx, err := types.SignTx(types.NewTx(&types.ConfidentialComputeRequest{ ConfidentialComputeRecord: record, ConfidentialInputs: confidentialDataBytes, }), signer, privateKey)

## Feedback Request

We encourage feedback from various stakeholders in our community, including SUAPP developers, wallet providers, and front-end developers. Your input is crucial for refining and enhancing this proposal.

For Suapp Developers:

- How do you see this change affecting your current workflow with CCRs?
- Are there any specific features or improvements you would like to see in the EIP-712 integration?

For Wallet Providers:

- Will this change simplify the integration process for signing and sending CCRs?
- Are there any compatibility concerns or additional support needed for EIP-712?

For Front-End Developers:

- Does the move to EIP-712 make building and maintaining interfaces for CCRs easier?
- Are there any challenges or areas where you need more documentation or examples?

Your feedback is invaluable in ensuring this feature meets the needs of our community.