EVM Blockchain Integration

General

The idea is to generate, store, and validate Agreement Proofs on the blockchain instead of IPFS. This will bring several benefits to the system:

1. Longevity: While IPFS data may disappear if no nodes are interested in storing the data, on the blockchain, the data is immutable and will remain forever.
2. Immutability: The data on blockchains like Ethereum or Polygon is much more difficult to tamper with than the data on IPFS.
3. Publicity: On the blockchain, all data and programming code are public. This will enable users to trust the proofs created on the blockchain as the logic of creating the proof, along with the proof itself, is publicly available. Furthermore, all signatures can be verified on-chain.

Store Proofs Metadata

General

A smart contract will be created to store metadata related to the Agreement File/Sign Proof and its version. This metadata will then be utilized in other smart contracts to generate proofs and verify user signatures.

Agreement File Proof Metadata

```
{
  "types": {
    "EIP712Domain": [
        { "name": "name", "type": "string" },
        { "name": "version", "type": "string" },
        { "name": "chainId", "type": "uint64" },
        { "name": "verifyingContract", "type": "address" }
    ],
    "Agreement": [
```

```
      { "name": "from", "type": "address" },
      { "name": "agreementFileCID", "type": "string" },
      { "name": "signers", "type": "Signers" },
      { "name": "app", "type": "string" },
      { "name": "timestamp", "type": "uint64" },
      { "name": "metadata", "type": "string" }
    ],
    "Signers": [
      { "name": "address", "type": "string" },
      { "name": "metadata", "type": "string" }
    ]
  },
  "domain": {
    "name": "daosign",
    "version": "0.1.0"
  },
  "primaryType": "Agreement"
}
```

Agreement Sign Proof Metadata

```
{
  "types": {
    "EIP712Domain": [
        { "name": "name", "type": "string" },
        { "name": "version", "type": "string" },
        { "name": "chainId", "type": "uint64" },
        { "name": "verifyingContract", "type": "address" }
    ],
    "Agreement": [
      { "name": "signer", "type": "address" },
      { "name": "agreementFileProofCID", "type": "string" },
      { "name": "app", "type": "string" },
      { "name": "timestamp", "type": "uint64" },
      { "name": "metadata", "type": "string" }
    ]
  },
  "domain": {
    "name": "daosign",
    "version": "0.1.0"
  },
```

```
  "primaryType": "Agreement"
}
```

Store Proofs

General

A smart contract should be created to store Agreement File Proof data, as well as all Agreement Sign Proofs data. This contract should be able to compute IPFS CID, which would allow for the generation of Agreement File Proof and Agreement Sign Proof. However, it should be noted that this contract does not provide proof verification.

Agreement File Proof

Store only message and signature

```
{
  "message": {
    "from": "<Creator's address>",
    "agreementFileCID": "<Agreement File CID>",
    "signers": [
      { "address": "<Signer 1 address>", "metadata": "{}" },
      { "address": "<Signer 2 address>", "metadata": "{}" },
      { "address": "<Signer 3 address>", "metadata": "{}" }
    ],
    "app": "daosign",
    "timestamp": <timestamp in seconds>,
    "metadata": "{}"
  },
  "sig": "<User's signature of Agreement File Proof Data>",
}
```

Agreement Sign Proof

Store only message and signature

```
{
  "message": {
    "signer": "<signer's address>",
```

```
    "agreementFileProofCID": "<Agreement File Proof CID>",
    "app": "daosign",
    "timestamp": <timestamp in seconds>,
    "metadata": "{}"
  },
  "sig": "<signer's signature>"
}
```

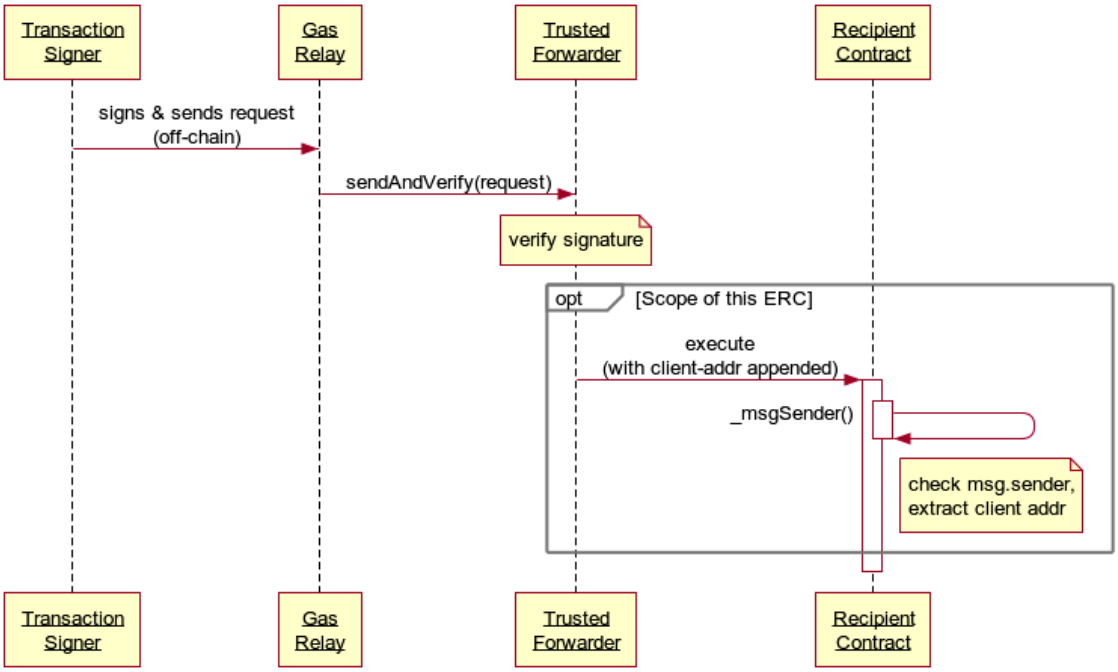
Validate Signatures of the users

General

Create a Solidity library to validate user signatures. The signatures can be either proofs that are stored on or off-chain, or simply signatures of general-purpose data.

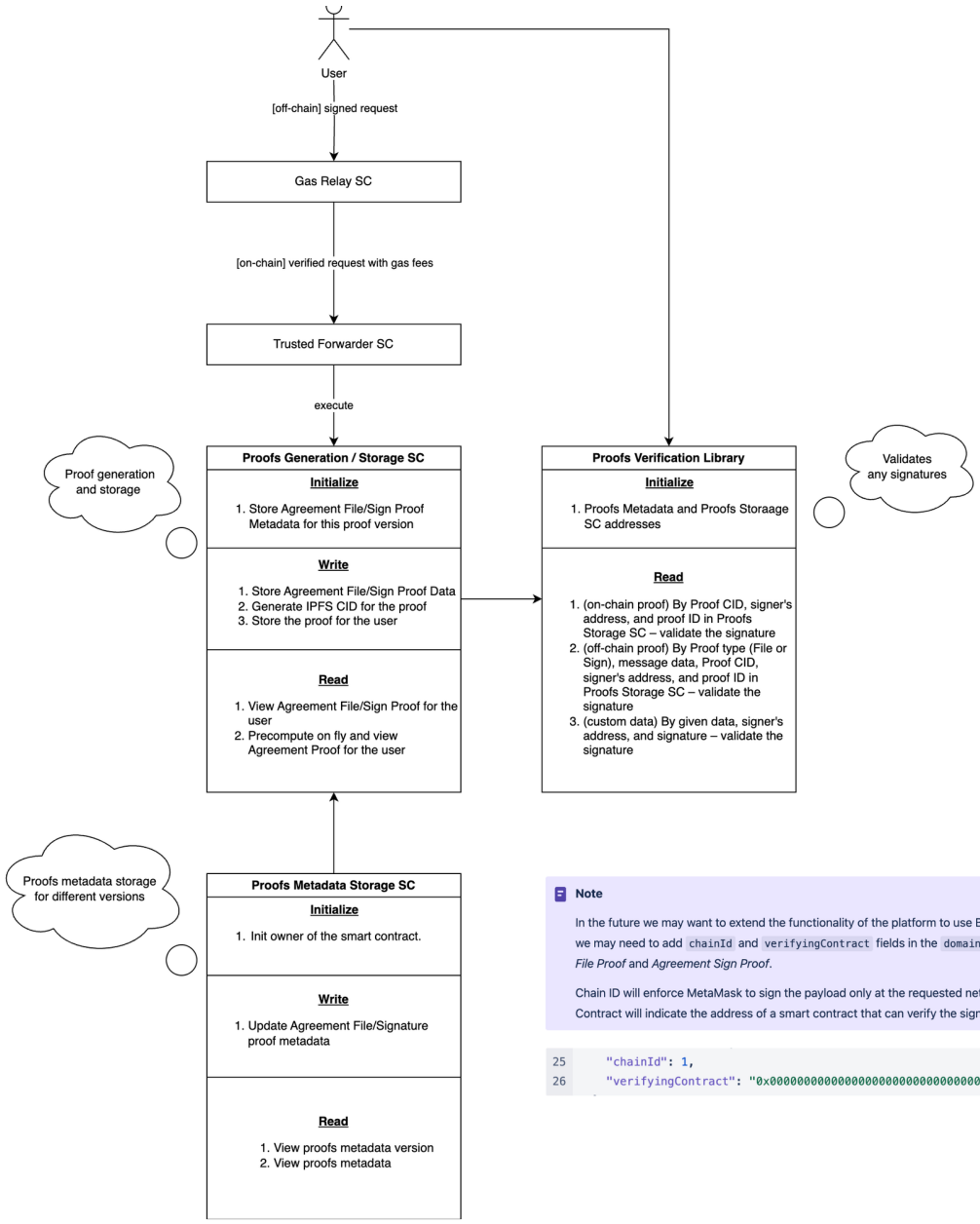
Gasless transactions for the end users

To send gasless (or meta) transactions, the EIP-2771 is utilized.



Technical Design Diagram

A tentative architecture for the smart contracts has been presented in this diagram:



Note

In the future we may want to extend the functionality of the platform to use Blockchain. In this case we may need to add `chainId` and `verifyingContract` fields in the `domain` block of *Agreement File Proof* and *Agreement Sign Proof*.

Chain ID will enforce MetaMask to sign the payload only at the requested network, and Verifying Contract will indicate the address of a smart contract that can verify the signature.

```
25      "chainId": 1,
26      "verifyingContract": "0x0000000000000000000000000000000000000000"
```