Feasibility Study - Data Availability with Validation on Cardano

Sky Protocol Team

2024-09-30

Overview

> Sky Protocol is committed to constructing a tailored Data Availability Solution aligned with the requirements of Cardano. Our approach involves publishing data in a format conducive for Cardano smart contracts to seamlessly validate and integrate into Layer 1 contracts. Rooted in Cardano and underpinned by its own Cardano-based token, our network reduces reliance on external networks and minimizes trust assumptions. Our network enables multiple Cardano-based DApps to leverage the same Data Availability network, thus pooling resources and capital for network validation.

This report addresses Project Catalyst milestone 1 of the Sky Protocol: Data Availability for Cardano Layer 2 Solutions proposal.

Criterion 1.a: Maintenance of Unique Identifiable Contract State

> How to maintain a unique identifiable contract state from one eUTXO to the next such that other contracts can read that state. This service may or may not already be provided by some Cardano contract languages, and may or may not be easy to provide on top of those languages. Presumably, underneath it all, an NFT identifies its holder as having “the” state for the bridge contract, and all lock scripts involved track which eUTXO possesses the NFT and what data is associated with it.

Results

Sky Protocol is a Data Availability (DA) Network. Its presence on-chain on Cardano is embodied in (a) a Bridge contract, allowing Cardano client contracts to see the contents of the DA, and (b) client contracts that can read the state of the bridge. A typical client will verify that some data was indeed published on the DA, based on which it will update the state of a side-chain or close a contested state channel.

For client contracts to be able to identify which eUTXO represents the state of the Bridge contract, a special Bridge NFT will be issued at contract creation, that will be held by the bridge. This is similar to how Oracles are implemented in the example from the Plutus Pioneer Program week 06. Indeed, a bridge can be seen as an oracle for the state of another blockchain — in this case, the Sky Protocol DA. Or more precisely, a complete bridge is made of two “lanes”, each lane being a bridge contract on one chain that serves as an oracle for the state of the other chain.

When a client contract needs to read the state of the Bridge, it can use the Bridge eUTXO, as identified because it is the one and only one holding the Bridge NFT, and include it as a **reference input** for the client transaction, as per CIP-31. This enables for arbitrary client code, without the Bridge itself having to include any explicit logic to support clients, beside exposing its state in the eUTXO.

Off-chain code will be used to find the Bridge eUTXO as the one holding the Bridge NFT; client code will verify that indeed, the reference input used as Bridge eUTXO contains the Bridge NFT.

The Bridge eUTXO will contain the Merkle root hash for a data structure that specifies all there is to know about the state of the DA: the public keys for the current committee, a trie of the current state of topics, a trie with the history of data published on the DA.

Criterion 1.b: Multisig for Data Availability Committee

> How to maintain a unique identifiable contract state from one eUTXO to the next such that other contracts can read that state. This service may or may not already be provided by some Cardano contract languages, and may or may not be easy to provide on top of those languages. Presumably, underneath it all, an NFT identifies its holder as having “the” state for the bridge contract, and all lock scripts involved track which eUTXO possesses the NFT and what data is associated with it.

Results

One crucial feature of Sky Protocol is the ability to rely on decentralized committees such that a 2/3+ε majority of honest functioning participants is enough to keep the system running. Thus, our Bridge contract must accept some M-out-of-N multisignature scheme, where N will be the size of its committee, and M will be 1+floor(2*N/3).

Initially, we will use a simple M-of-N multisig based on explicitly counting Ed25519 signatures. This is readily possible with a Plutus script, using Plutus builtin functions that allow one to check signatures of arbitrary data against public keys themselves provided as data. Note that this is different from pre-Plutus support for multisig, that only supports signing the entire transaction against a set of public keys known in advance. Also note that the size of the committee is limited by the necessity to publish the public keys and signatures on-chain in a 16384-byte transaction; the exact size limit will depend on the contract as finalized, but from an estimate of data sizes (32 byte for public key, 64 for signature) means that the maximum committee size would be a bit over 100.

In the long run, we can use BLS or other efficient multisig schemes based on pairings, thanks to CIP-0381, that was specifically added for this purpose. We can even have two or more sets of public keys: one for use in the “happy case” where everyone cooperates (N-out-of-N signature), that is cheaper to verify (single combined public key) one for use in the “unhappy case” where some committee members fail to cooperate, that is more expensive to verify (actually N public keys to check individually) but ensures the system survives bad episodes. There could even be intermediate signature schemes allowing for several levels of progressive performance degradation. This choice of multiple algorithms will not change significantly the size of committees.

One final technique we could use is Zero Knowledge proofs (ZK proofs). A ZK proof is much more expensive to create or verify than the other signatures. However, (a) it can scale to much larger committees, and (b) the technique can be used to validate more about state of DA updates than just signatures. In the long run, we will want to use such ZK proofs; but in the short run, we can and should launch Sky Protocol with simpler multisig techniques that are easier to build and deploy.

As part of the Merkle root hash associated to the Bridge eUTXO (see section above), the Bridge will commit to a committee as specified by its sets of public keys for each of the supported multisig algorithms. The verification can proceed from this set of public keys.

Criterion 1.c: Merkle Proof of Data Availability

> How to verify in some smart contract language a “merkle proof” that data was present in a merkle tree the root of which was signed.

Results

Both the Bridge contract itself and the client contracts will want to verify parts of the DA contents as signed by the Sky committee. A signature commits to a simple 32-byte datum, the hash of a larger data structure. To verify nested fields of that data structure, we will implement the usual Merkle membership proof approach in Plutus. We may reuse existing libraries such as Plutus.MerkleTree, or reuse the implementation techniques underlying such libraries.

The path segments from the root to the leaf nodes of interests are included in the proof, together with the hashes of the parts of the tree not visited. Only the root hash is signed, its signature verified.

Conclusion

The techniques we set out to use are all well-known, with documented examples in the Cardano ecosystem: NFT-based oracles for contract state, M-of-N Ed25519 multisigs, Merkle membership proofs.

Therefore, we are confident that the contract we set out to write is feasible.