Recently, I've been thinking about encoding standards, particularly those which we will want to standardize soon for application-facing components of the protocol such as the resource machine - relevant data structures here include resources, transactions, and data processed therein (such as authorizations which are signed).

We don't want to reinvent-the-encoding-wheel, so one option is to use an existing standard such asprotobuf, RLP, or SSZ. It's not clear that these formats provide exactly what we need, however. I think our needs are roughly:

- typed serialization (unlike RLP or JSON) - more efficient and we need to typecheck anyways
- supports basic types + recursive product/coproduct types
- efficient to deserialize / check deserialization in a circuit
- defines canonical Merkle root of every object
- as simple as possible given these constraints

One interesting option which I've been considering is not

to define an encoding scheme at all, but instead only to define a Merkleization one. In particular, consider the following scheme:

Define an ObjectType

as either:

- a basic type (such as a bytestring or unsigned integer)
- a product type (of object types)
- a coproduct type (of object types)

i.e. (in Haskell, assuming a basic type BasicType

)

data ObjectType = BasicType BasicType | ProductType [ObjectType] | CoproductType [ObjectType]

and, similarly, ObjectValue

data ObjectValue T = BasicValue T | ProductValue [ObjectValue T] | CoproductValue Nat (ObjectValue T)

where the natural number index indicates which coproduct "slot" is occupied.

We can then define the following recursive Merkleization function, assuming hash

:

root :: ObjectValue T -> ObjectType -> ByteString root (BasicValue b) (BasicType*) = hash b root (ProductValue xs) (ProductType*) = hash $ concat (hash 0 <> map hash xs) root (CoproductValue index x) (CoproductType _) = hash $ hash (x + 1) <> hash x

Represented visually:

[

excalidraw-merkleization (1)

2701×2013 269 KB

](https://europe1.discourse-cdn.com/standard20/uploads/anoma1/original/1X/281af7f1e9d202a89203fcfdc8b1338fc910969e.png)

IF we simply:

- Send around Merkle roots of each object, and
- Send any part of the data for that object (encoded in any fashion)

THEN - I think - we do not need to standardize any further encoding at all! All we need to do is send around Merkle roots for all objects and check - when deserializing or processing objects - that the data provided matches the Merkle root. Naively implemented, this would incur a lot of overhead (calling hash many times), but this overhead is probably not meaningful compared to the data storage and bandwidth costs, we can cache to avoid recomputation, etc.

This Merkleization scheme also has several nice features:

- it’s possible to verify constant-size parts of a large object with only constant-size data
- product type Merkleization can be parallelized
- it’s very simple

We will still need to standardize encoding for basic types with proper domain separation (distinct values for different types should not have the same hash), but this can probably be borrowed from existing work.

cc @mariari @Moonchild @vveiln