I am interested in discussing ABI creation. More specifically the potential to standardize ABI creation to the point where ABIs become deterministic and as such are in a canonical form. Once this is achieved, we can easily perform hashing to create unique handles for ABIs.

Solidity

The order of functions in Solidity smart contract source code does not

have an effect on the order of the items in a Solidity generated ABI.

Vyper

In contrast, the position of variables, functions and events in Vyper source code does

have an effect on the position of the variables, functions and events in the Vyper generated ABI.

EWASM

Please comment …

There is a more elaborate example of the above (with source code and ABI snippets)here if you are interested.

What are your views on reproducible builds (deterministic compilation) in the context of Ethereum ABIs? Is it fair to say that deterministic compilation is more to do with the bytecode and less to do with the ABI (being that the ABI is not executable).

I personally see value in having deterministic ABIs because they are the key to storing/organizing and discovering contracts on the network. I am currently using a web3.toHex(web3.sha3(text=_abi)) approach to create a deterministic ABI hash in canonical form (at present by leaving the top level ABI functions as per the Solidity compiler output, but in addition, by sorting the (natively unsorted) internal objects and lists of objects before creating the hash).

For example, this web page lets you search for contracts using an ABI as the search query

I noticed (after writing this post) that @axic has a suggestion of deterministically sorting lexicographically by values of keys I think this makes sense.