Background

In ERC20-based DeFi workflows, users interact with application contracts (e.g. Uniswap) that then invoke token contracts (ERC20) to transfer tokens on users' behalf. Due to the layer of indirection, an additional approve

call is required.

There are has been numerous proposals to amend and improve upon ERC20, such as ERC223 and ERC677.ERC677 proposes transferAndCall

, which changes the workflows of current ERC20-based protocols, in that users would interact with token contracts instead of application contracts. It also complicate workflows that require multiple input tokens (e.g. liquidity pool deposits).

A better authorization method via access to CALLDATA of non-current call frames.

This thread is an open discussion around an alternative authentication mechanism via CALLDATA

. The idea originated in the work ofFLAX in the context of composable privacy for Ethereuem-style DeFi but can be considered independently.

Below is an initial sketch if access to transaction CALLDATA

is supported.

1. Add opcodes for accessing CALLDATA of the user transaction in EVM, tentatively ORIGIN_CALLDATA{LOAD,SIZE,COPY}

, and provide support for it in Solidity via tx.msg

.

1. Standardize authorization message format inside tx.msg

. The message will authorize one-time

use of a user's token to a contract. For example, each authorization could include the token address, spender contract address, and amount authorized. And each transaction could include multiple authorizations.

1. Token contracts must

verify authorizations in tx.msg

inside transferFrom(sender, recipient, amount)

, which amounts to checking that (1) tx.origin == sender

, and (2) tx.msg

authorizes recipient

to spend amount

of this

-tokens.

1. Generation and parsing of token-use authorization can be built into wallet implementations, alerting users of the side effects of their transactions.

More generally, the mechanism can also be used to support delegation of token use between contracts if CALLDATA of intermediate call frames are accessible via opcodes. In this scenario, the token contract will check, at step 3 above, that the there exists

a call frame where caller

is sender

and the CALLDATA

of the frame authorizes the spend.

This proposal require changes to the EVM and its surrounding tooling infrastructure. However, it preserves the current ERC20-based contract logic and can also used to support anonymous token standards as proposed in FLAX.