TLDR

: We suggest a way to bootstrap accounts into account abstraction schemes at the application layer.

Construction

We build an “abstractor” contract to emulate account abstraction at the application layer. The abstractor has “virtual accounts” serving as “virtual entry points” for “virtual transactions”. Each virtual account has a “virtual balance” and “virtual init code”.

To make it to their virtual entry points, virtual transactions are broadcast by users to proposers. Those are executed locally, selected, batched and finally wrapped by proposers in a single “dummy transaction” where:

- The sender address is a fresh ephemeral account with zero balance
- The nonce is set to 0
- The gas price is set to 0
- The gas limit is set to 8,000,000
- The recipient address is set to the abstractor’s address
- The value is set to 0

The EVM unwraps the dummy transaction and sends the virtual transaction batch to the abstractor. Individual virtual transactions are then processed sequentially as expected:

1. They are parsed by the abstractor to extract the associated “virtual sender address”, virtual gas price", “virtual gas limit”, “virtual data”, “virtual recipient address”, virtual signature", “virtual nonce”, etc.
2. The virtual signature and virtual nonce are checked against the virtual init code of the virtual sender account and the abstractor throws if the checks fail.
3. If the virtual balance is less than the virtual gas price times the virtual gas limit the abstractor throws, and the virtual balance is sent to the coinbase.
4. The abstractor calls the contract specified by the virtual recipient address with the virtual data, and a gas limit set to the virtual gas limit.
5. The abstractor rewards the coinbase with the gas used in step 4) times the virtual gas price.

Other details of account abstraction (e.g. init code filling) are similarly handled by the abstractor.

Discussion

The default signature and nonce infrastructure is bypassed by using an ephemeral account with 0 gas price and nonce, and a dummy signature. This makes it possible to have alternative transaction entry points (e.g. to support UTXOs, quantum-secure signatures, ERC20 gas) without being burdened by ECDSA signatures and nonces.

Pushing account abstraction to the application layer keeps the protocol layer simple. Simultaneously, different abstraction schemes can compete and evolve organically at the application layer, and explore the gamut of tradeoffs that a one-size-fits-all abstraction scheme may fail to capture.

Going further, we may be able to reuse the above bootstrap strategy on a transaction entry point that is simpler than Ethereum 1.0 accounts to remove enshrined infrastructure such as gas prices, nonces and ECDSA signatures, all without enshrining abstraction-specific infrastructure such as PAYGAS, BREAKPOINT or PANIC opcodes.