Account abstraction is a powerful idea that allows e.g. for a cleaner EVM, better multisig, post-quantum secure signatures, and transaction expiry. It turns out account abstraction is general enough to allow for “miner data”, and this is very helpful for auto-updating witnesses in the context of stateless clients.

I call “miner data” transaction data provided by the miner during block creation. Instead of being signed by the transaction sender, miner data is constrained differently by the account abstraction init script. Below are examples of miner data constraints:

1. No miner data

(minerData.length == 0

): This is how transactions are processed today.

1. Miner-updated witnesses

: From the stateless client proposal, “we put the witness outside the signed data in the transaction, and allow the miner that includes the transaction to adjust the witness as needed before including the transaction”. Using account abstraction for witnesses is cool for a couple reasons: * It means we don’t need to design an ad-hoc mechanism to include updated witness data, keeping the EVM clean.

- It means “fancy accumulator schemes” can be first-class citizens! See thediscussion here: “you can totally make your own contract that uses some fancy accumulator scheme, and it’ll be almost first-class (almost because you won’t get the benefit of miners auto-updating the witness if multiple actors send a transaction to modify the same accumulator at the same time)”
- It means we don’t need to design an ad-hoc mechanism to include updated witness data, keeping the EVM clean.
- It means “fancy accumulator schemes” can be first-class citizens! See thediscussion here: “you can totally make your own contract that uses some fancy accumulator scheme, and it’ll be almost first-class (almost because you won’t get the benefit of miners auto-updating the witness if multiple actors send a transaction to modify the same accumulator at the same time)”
- Miner-updated market data

: Market data (e.g. stock prices) included in transactions can become stale between when the transaction is sent and when the transaction is mined. Miner-updated market data could “fill the latency gap”, and improve upon expire-and-rebroadcast schemes that suffer from more coordination with the miner, and higher bandwidth and latency.

1. Miner discrimination

: It is possible to give exclusive mining rights for a transaction to a specific miner by having the init script require a signature of that specific miner in the miner data. This is possible today (broadcast your transaction to just one miner) but would add cryptographic guarantees.

1. Arbitrary miner data

: This may be useful, e.g. to add entropy.