The benefits of Midnight's model

A reasonable question might be why Midnight needs any notion of public state at all. Wouldn't a smart contract with no publicly visible data also achieve better confidentiality?

The need for public states

This is appealing, and in certain cases, better privacy can be achieved using additional cryptography, such as secure multi-party computation (MPC) or fully-homomorphic encryption (FHE). These approaches generally come with their own drawbacks, however, and there is one central reason that publicly visible data *is* a core part of Midnight's model: *it is often desirable.*

In particular, public data is essential in a decentralized system. Users of a decentralized system want to be able to join smart contracts freely, and contracts are often designed without knowing who will use them ahead of time. This *requires* some sharing of data – how do you know if you want to interact with a contract if you don't know what it is or *how* to interact with it?

The premise of Midnight is to enable seamless interaction between the shared public data of a contract and confidential data that you do not wish to share. For instance, an auctioneer wants to show off *what* is being auctioned and perhaps (depending on the auction) the current highest bid, while a winning buyer wants to keep their identity secret. Or, an insurance company may wish to list its poli publicly, while clients do not wish the details of their policies to be listed.

Ask AI

Contention

Feedback

One problem that can arise with public states is *contention*. If multiple users interact with a contract, naive designs can lead to the users stepping on each other's toes. Consider a simple counter contract, where users increment a publicly stored counter. A naive implementation may a) read the current value, and b) set the new value to the read value + 1. This can be a problem if the steps are recorded separately in the transaction, say as `[read 1, write 2]`. If two of these transactions get submitted simultaneously, they will conflict, and only the first will succeed. For the others, the value read is no longer `1`, so the transaction will fail. If instead the transaction is structured to contain a single-step increment – for instance `[incr 1]` – then all transactions can succeed.

Midnight is designed to help contract authors structure their interactions in a non-conflicting way, so that contracts do not need to deal with contention in most cases. In some cases, contention is unavoidable: If I put $10 in a pot, only the first person can claim it, by design.

Transaction fee predictability

One aspect still under revision in Midnight is the *predictability* of transaction fees. The design goals are both that users don't overpay for transactions and also that users don't pay fees for transactions that fail.

Midnight's on-chain language, Impact is designed so that users can reliably predict fees in many cases, and our approach to transaction fallibility enables smart contract authors to structure contracts so that failed transactions fail early, before fees are taken.

Feedback