(1c) - Hybrid Systems

There are a variety of system designs possible beyond the proof of work and authority sealing mechanisms we've explored here. Name one security problem with the proof of authority sealing mechanism you created above. Can you think of a technically sound approach that could help mitigate this problem (imperfect mitigations are fine)? (Please do not describe existing protocols, including but not limited to Proof of Stake protocols, Tendermint, or PBFT-based systems; originality is highly valued here!)

One problematic security aspect proof of authority sealing mechanisms is that because validators must have their identity formally verified to be approved, there no anonymity at all. While this might initially seem like a safer option than other systems, as there is more transparency and ownership to each validator and their work, this also means that such validators know who each other are and could find each other and create a system in which they are manipulating the transactions. The PoA system of attaching reputation to a validator's identity only prevents one single validator from diverging, but if enough validators join together and agree to ignore the risks of damaged reputation, this incentive to behave fairly and justly no longer exists.

References:

https://en.wikipedia.org/wiki/Proof-of-authority

https://ethereum.stackexchange.com/questions/13736/what-are-the-limitations-of-proof-of-authority/19457#19457

Problem 2 - UTXO Management in Wallets

In Lecture 6, we learned about the representation of Bitcoin as inputs and outputs. Additional reading on this "UTXO" model of representing account state is available in the textbook. With Cornell Chain, we have provided some code that generates a random proof-of-work blockchain; in this code, we simulate various users transacting with wallet software, choosing which UTXO to use randomly from the UTXOs available to a given user. Unfortunately, this strategy is clearly not efficient: wallets want to maintain as few UTXOs as possible in Bitcoin, since transactions are charged according to size, and using / creating more output costs money.

Consider an enterprise business called Moonbase, which must make and receive hundreds of thousands of transactions to millions of users every day, paying millions of dollars in network fees. Moonbase's strategy is as follows: for every user that wants to deposit money, Moonbase receives a new UTXO from (and created by) that user. For every user that wants to withdraw money, they create a withdrawal transaction funded by a random UTXO that is large enough to fund this transaction.

1. **Identify a denial-of-service vector in this process, i.e., an adversarial attack strategy by dishonest and wealthy users that could result in failed withdrawals for legitimate users of Moonbase. Provide a fix for this DoS vector and argue informally that user withdrawals will never fail.**

One adversarial attack strategy could be if the dishonest and wealthy users banded together (they can communicate between each other) and sent massive amounts of transactions consisting of very small amounts to flood the block maximum. Since they are wealthy, they could easily cover the transaction costs. This enormous amount of resulting UTXOs, all representing tiny amounts, would mean that legitimate users of Moonbase would be prevented from withdrawing their funds because there would be way too many UTXOs of small amounts that wouldn't be large enough to fund any of their transactions. One potential fix would be a floor minimum for the amount a user could transact, rendering the playing field fairer to those users who are using the platform honestly and legitimately. Therefore, the adversarial attackers wouldn't be able to send tiny amounts (such as 1 cent) to flood the block.

Another attack strategy would be for a very wealthy group of users to manipulate the transaction fee and raise it to a high amount that would preclude normal users from affording such fees. If I wanted to pay a transaction fee of 10x the usual amount, my transaction is more likely to go through. If I continued to pay this 10x transaction fee, the other users wouldn't get their transactions through, as they are only willing to pay a fraction of this. A fix to this situation would be a set, immutable transaction fee cost that wouldn't allow these wealthy bad guys to manipulate and increase the transaction fee to exclude the legitimate user withdrawals.

2. **For each user withdrawal in the provided scheme, recall from class that two UTXOs actually need to be generated: one paying the target user, and one that is kept by Moonbase representing any leftover "change". Provide a modification to the above strategy that will reduce the number of UTXOs Moonbase must maintain in its database.**

One simple modification would be to consolidate all one single user's UTXOs so that Moonbase wouldn't have multiple UTXOs per user based on their number of transactions, but instead one single UTXO per user. Therefore, if Alice has 10 Bitcoin and sends Bob 5 Bitcoin, one UTXO would go to Bob (5 Bitcoin), and the other would be stored by Moonbase (5 Bitcoin). If Alice sent Carl 3 Bitcoin, then one UTXO would go to Carl (3 Bitcoin), and Moonbase would add this new UTXO of 2 Bitcoin to Alice's first UTXO, giving her one single UTXO that Moonbase must maintain. This system will also have the benefit of ensuring that not all UTXOs will represent a tiny value, which is problematic when a UTXO has to be randomly selected to fund another user's transactions and there are not enough large UTXOs. In addition, not all users will be making the same volume of transactions, meaning that there will still be some users with small-valued UTXOs available for similarly small withdrawals.

Alternatively, Moonbase could require that all users consolidate their UTXOs themselves on a regular basis (say monthly). In this case, a user would collect all their UTXOs consisting of change and sent it to themselves in one single transaction. This would create one single output for any future transactions, and Moonbase wouldn't have to maintain many tiny UTXOs per person.

References:

https://www.youtube.com/watch?v=IehHwBt3NAg

https://medium.com/@lopp/the-challenges-of-optimizing-unspent-output-selection-a3e5d05d13ef

https://www.reddit.com/r/Bitcoin/comments/7rxiwh/few_thoughts_about_utxos_consolidation/