AVOUM: Account-View-on-UTXO-Model For Cardano Project Catalyst F5

A project by Mutual Knowledge Systems, April 2021
https://j.mp/AVOUM_F5

DeFi smart contracts that anyone can interact with are subject to Denial-of-Service (DoS) attacks whereby the attacker constantly races other participants to the current UTXO. We propose a solution to this issue by offering an “Account” view on top of the UTXO model of Cardano, using malleable transactions and Node-Extractable Value. Thus equipped, the Cardano blockchain can offer the best of the Account model and the UTXO model, beating rival blockchains on all counts.

1. Introduction

1.1. An Auction gone wrong

Let's imagine that Alice wants to sell a painting by her famous deceased father in an online auction. She chooses to do it through a contract on a blockchain that uses the (E)UTXO model. The painting is worth about one million dollars, which is what the highest bidder, Bob, is ready to offer. Bob signs a transaction to purchase the painting, and waits… but instead the painting ends up being awarded to Mallory, who only paid Alice a few cents for it! What happened?

To interact with the auction contract, Bob's transaction had to refer to its current state, embodied in a UTXO. But what if some other user also posted a bid, that appeared before Bob's? The bid might be less than Bob's, but because it appeared before Bob's, it interacted with the contract, consuming its previous UTXO and generating a new UTXO that accounts for that new bid. Bob's transaction, referring to the old UTXO, is therefore invalid, and Bob has to sign a new transaction and wait for it to be either confirmed or rejected. But what if yet another bidder managed to get their bid in before Bob's? Then Bob must try again, in an endlessly repeated cycle, until the end of the auction arrives and his bid never makes it through.

What did Mallory do? She sent to the blockchain nodes a stream of transactions making bids for one cent, two cents, three cents, etc., such that the UTXO constantly changes, and changes faster than Bob or any other honest but technically unsophisticated bidder can react. Regular users wait for a couple of blocks each time to determine what the UTXO to sign a transaction against will be, but Mallory makes multiple transactions at every block, and even maintains multiple rival states of the contract so the odds that honest bidder write their transactions against a valid UTXO are very low. Mallory also watches how much Bob puts down in transaction fees, and makes sure to always leave a larger tip to the nodes, so she'll be sure that they prefer her transaction to Bob's. In the end, Mallory racks up a very large bill in transaction fees: thousands of dollars worth of node tips, maybe much more, along the duration of the auction. But in the end, she manages to keep out all the other bidders, and win the auction with a ridiculously low bid. For a few thousand dollars in fees, she wins a million dollar worth asset.

Mallory can similarly run a *Economic Denial-of-Service Attack* (Economic DoS) to exclude rivals from any contract such that which she could benefit from this exclusion.

1.2. A Market for Open-Contract Transactions is Inevitable

Any "open" contract on a UTXO blockchain can be subject to the same Economic DoS: if anyone can interact with the contract, or if an existing participant can interact with it an indefinite number of times before others may react, then a sophisticated attacker can use the same technique as described above to effectively prevent any other unsophisticated participant from interacting with the contract until after it is too late.

Even without an intentional attacker, the UTXO for heavily-used contracts will be subject to heavy contention and frequent change, such that regular non-technical users will have trouble interacting with it. They have to watch the blockchain, determine the UTXO for the contract, sign a transaction with this UTXO, wait for this transaction to either make it through or be invalidated by a rival transaction, then sign a new transaction and try again, repeating for hours on, hoping that at some point there will be a lull in the activity of that contract and their transaction will finally make it through.

Does that mean that open contracts on UTXO blockchains are forever reserved for use by sophisticated participants who can run a full node, watch the blockchain at high-speed, and race rival transactions in all of multiple possible worlds at the same time, in servers close enough to the nodes to matter?

Actually, no: *someone* sophisticated has to do this job. But if the open contract is properly structured, then this *someone* can be any participant in a *transaction posting auction market*. Then, the economically competent but technically unsophisticated user posts only have to pay a sufficient transaction fee, and the sophisticated professionals will compete with each other to be the one who effectively posts the transaction.

A single sophisticated professional — offering his services to ensure that users' transactions get through in a timely fashion to interact with frequently changing contracts — would likely be enough to start this market. Posting a transaction then becomes fire-and-forget for regular users, and the professional will earn high fees for consistently winning the race to the UTXO. High fees will likely attract competition, until a market emerges where sophisticated users compete to earn fees from users, driving those fees down, and flooding the network with rival transactions.

Eventually, nodes would realize that they could cut out the middleman between end-users and the blockchain by running those servers directly on their mining rigs, saving a lot in fees and network traffic for everyone.

In the end, when the market finally reached its equilibrium, users would interact with open contracts by posting suitably malleable transactions to the blockchain, and trust that nodes would automatically substitute the most recent contract UTXO for the one they posted, thereby having the blockchain behave exactly as if it had an Account/Balance model.

1.3. Our Approach

We propose to directly implement the final equilibrium state of this process, wherein nodes implement the Account/Balance view of open contract on top of the underlying UTXO model, thereby providing the best of both worlds. Short-circuiting this market evolution allows us to simplify away any intermediate API, and go straight to a simplified version that only requires additional processing by the nodes, with no additional API.

We thus propose to implement for the Cardano blockchain:

1. A convention for contracts to accept signed malleable transactions that do not depend on the specific UTXO of the contract, with a Plutus library to implement the convention,
2. An algorithm for nodes to recognize the convention, detect what validity conditions exactly a signed transaction does or does not depend on, and automatically re-run an updated transaction with a modified contract UTXO when appropriate.
3. An optimization heuristic for nodes to optimize their earnings based on the available pool of potential transactions.

2. Background Concepts

2.1. UTXO Model vs Account/Balance Model

Cardano, like the first blockchain, Bitcoin, represents available assets as UTXO, which stands for Unspent Transaction Outputs: each transaction may take as inputs some UTXOs of previous transactions (which are therefrom spent and no longer UTXOs) and itself produces UTXOs, thereby transferring the assets from previous owners to new ones. Ownership is enforced by the “locking script” associated with each UTXO, that ensures that only the possessors of some cryptographic keys may spend the UTXO, either freely or according to some “covenant” that limits their ability.

By contrast, the first blockchain with the ability to write smart contracts, Ethereum, uses an Account/Balance model, transactions transfer assets between accounts that persist across transactions, by modifying their balances. Some accounts are simply controlled by cryptographic keys, while more elaborate accounts are controlled by “smart contracts” which are scripts written in the blockchain’s virtual machine.

UTXOs make it easy for participating network nodes to verify historical transactions, in parallel—whereas the Account/Balance model, at least in its naive implementation, makes playing or replaying historical transactions an essentially sequential activity. Joining the network and validating its state can therefore be done faster with UTXOs than with Account/Balance, for the same reason. UTXOs provide an all over more robust data model.

On the other hand, the Account/Balance model makes it much easier to interact with “open” contracts involving a lot of participants: a user can “just” sign a transaction that describes the action they want to take, and this transaction will be valid irrespective of the state of the contract when the action is taken. By contrast, with the UTXO model, the user would have to track down exactly the state of contract (its UTXO) at the time the action is to be performed to even prepare the correct transaction; but this state could change very quickly, which opens the contract users to the Economic Denial-of-Service Attack described above.

2.2. FOMO3D and the Dark Forest

Economic Denial-of-Service Attacks are not mere speculation. And they are not just for UTXO blockchains. Indeed, they already happen, all the time, on the Ethereum blockchain, that uses the Account/Balance model! The most famous stories about it are the winning of the FOMO3D tontine, and the Dark Forest of front-runners for swap contracts.

The FOMO3D was a joke of a contract, a blockchain variant of a *tontine*. In a historical tontine, many participants, usually young people, put money in a pot, or left it to a careful money manager—until all the participants died but one, who then got all the money in the pot. In times and places when life expectancy was shorter than the modern world, the lone survivor might even still be young enough to actually enjoy his fortune. To adapt the concept of tontine to the blockchain, a simple mechanism was used to detect the *last man standing* (or then again last robot operating): anyone could add tokens to the pot at any time (above some floor contribution) and thereby become a participant; to take the tokens out, you just had to be the last one putting tokens in, with no one else having put in any tokens for the last 24 hours. Of course, some rival people were running robots to prevent anyone else from winning, by making sure to chip in a few tokens after 23 hours and 55 minutes or so, if needed. And yet, some clever person managed to steal the then million-dollar pot while rival robots were still operating. How did he do? By running an Economic Denial-of-Service Attack: he simply put money in the pot, waited for about 23 hours and 55 minutes, and then bought the entire Ethereum blockchain for 5 minutes until his rivals had been excluded long enough for him to take the money out. How did he buy the entire blockchain? With repeat copies of a transaction that paid for the entire block gas limit at a gas price advantageous to the nodes, until the 5 minutes had passed.

In the Dark Forest story, also on Ethereum, a person noticed a way that anyone can take tokens out of a misconstrued ERC20 token contract with a clever swap transaction. A good actor, he decided to take the tokens out before they disappear, and safeguard them for their original owners. But there are robots out there that watch all ERC20 contracts for any advantageous swap transaction and will front-run that transaction with a rival transaction executing the same swap, but to the advantage of the robot operator rather than the original arbitrageur. To make sure the rival transaction makes it in, the robot simply puts a larger gas price, as long as there is any profit in the swap large enough to justify the gas cost. And our good actor was thereby scooped by a robot, and concluded that Ethereum is a Dark Forest where you'll easily get mugged.

2.3. The Implicit Auction for Blockchain Space

Thus, Economic Denial-of-Service Attacks already exist, and so even on Account/Balance blockchains. At the bottom of the issue, the underlying reality is that *space on the blockchain is scarce*. Whether limited explicitly through a conventional block size limit, a conventional gas limit, or just implicitly via the network bandwidth of the winning nodes, there is a limit on how many transactions can make it to any given block. From an economic point of view, that means that *getting a transaction into a block is winning an auction*.

Therefore, the first issue, demonstrated by the FOMO3D story, is that of technical sophistication. When one participant knows how to play the transaction-posting auction correctly, when the other participants are too inept to even realize the game they're playing,
more technically sophisticated than others, they the sophisticated participant can run circles around the other ones and prevent them from ever getting their transactions to the blockchain, then causing their rivals to timeout and extracting any resulting value from the smart contract.

In the case of racing a UTXO, our solution above makes the same sophistication available to every participant, for a fee. Sophisticated professionals, typically nodes, handle the complexity for regular users, and the technical problem has been reduced down to the underlying economic problem. In the case of the FOMO3D story above, the economic problem was hidden under a very thin layer of technical sophistication. In both cases, the auction for blockchain space, previously left implicit, was made explicit: to get your transaction in, you need to bid a higher fee than your rivals—which shouldn't be a big problem if you're playing a positive sum game while they're playing a zero-sum game.

Now, the auction for blockchain space is a *first price auction*, since a second price auction would be gameable by nodes. Thus, to win an auction and get their transactions in, the participants must remain active, monitor the blockchain, and carefully increase their bid with market conditions, alongside their rivals; otherwise they'll pay too much (if they reveal their actual maximum bid too early), or they'll fail to get in (if they fail to raise the price to their maximum bid). There is still *some* sophistication required on the side of the participant: not technical sophistication, but economic sophistication. Posting transactions is still not fire-and-forget: users must keep watching the blockchain, check whether their transaction went through, and if not, sign a new variant of the transaction with a higher fee and watch again (and be ready to be surprised in case their old transaction made it through after all). This matters a lot whenever there is a deadline to sending transactions, which is almost always the case when writing robots that automate participation to some DApps.

To go back to our illustrating example, the FOMO3D story, "all" the tontine-participating robots had to do to keep playing their chicken game was to match and slightly outbid the fees posted by the attacker for one block. Indeed, to win the pot, the attacker had to win the auction for each and every block for the entire duration of the attack, whereas the defenders would only have had to win one single block. Since there were about 20 blocks in those five minutes, they could have let him buy fifteen entire blocks before they outbid him, making it a dear lesson no would-be attacker would forget. Defending successfully would have been an order of magnitude less costly than the attack. It would have been yet another order of magnitude less costly if they had started their transactions an hour in advance rather than waited until just five minutes before the deadline. Or if the blockchain had had ten times as much throughput. In general, defenders against Economic DoS attacks are at an advantage and attackers are at a disadvantage—but only if they play the game correctly. With economically rational and technically correct robots, the attacker would stand no chance. But that in itself is the topic for a separate project.

2.4. Node Extractable Value

The second issue, demonstrated by the “Dark Forest” story, is that in a sophisticated-enough market, the nodes will extract the value that is theirs. And this value comprises the fees that a transaction is worth paying for, as well as any profit that can be made by rewriting malleable transactions to their profit. This includes changing the name of the beneficiary of a single-transaction arbitrage—a zero-sum game where the user loses—as well as inserting the latest UTXO in a transaction to extract a fee—a positive-sum game where the user wins. In doing so, they are not being evil in one case and good in the other, they are just playing the game competently, as designed.

Super-competent nodes might even loosely collude with each other to squeeze higher transaction fees from users who fail to tip their transactions sufficiently when they have a short deadline. This collusion in practice will be counterbalanced by the interest of the node to defect from the cartel and accept the lower fee for themselves, rather than help another node collect the higher fee. But the more concentrated the mining power, the more the nodes should cooperate to raise fees. However, this is speculation for a distant future.

Our present project is to build the engine for nodes to play the positive-sum game where nodes and users win as the users’ transactions get posted.

3. Work to be done under this Proposal

For the purpose of this proposal, we estimate that each full-time qualified hire will cost MuKn approximately $15,000.00 per month, that each half-time qualified hire will cost $7,500.00 per month, and each quarter-time qualified hire will cost $3,750.00 per month.

3.1. Step 1, for Catalyst Fund 5: Proof of Concept for Malleability in Plutus

The first step for this project is to write a simple “open” contract, in a style that allows for malleable transactions where a UTXO can be substituted for another. The work can be described as follows:

1. Implement a contract for a simple first-price open-bid auction.
2. The contract shall be implemented on top of the Plutus DApp framework, or as small a slight modification thereof as necessary.
3. The contract transactions should be malleable so that it remains valid when the UTXO changes because someone else made a bid.
4. We will provide a function to tweak the posted transaction to adapt it to a new UTXO for the “same” contract.

Deliverables:

1. If possible, (a) a working contract as described above and (b) a corresponding transaction tweaking function, with (c) an example execution of the previous.
2. If the above is not possible due to current limitations of Plutus, we will deliver a set of proposed API modifications that would make the desired contract possible.

We estimate this step as taking about six weeks for one full-time developer, one quarter-time architect and one quarter-time manager, for a total budget of $45,000.00 including taxes and overhead.

4. Future Steps for *subsequent* Catalyst Funds:

4.1. Step 2: Proof of Concept for Malleability Support in Node

The second step will be to develop a change to the node software so it can correctly run the auction contract.

During this step, we will likely identify changes to be made to the node software that will be required to properly take advantage of transaction malleability in the node. In this case, the deliverable shall be not the actual support for the auction contract, but a set of proposed API modifications that would make this support possible.

4.2. Step 3: A Library for Open Contracts in Plutus

The third step will be to develop a general-purpose library for “open” contracts in Plutus.

Assuming that any required changes to Plutus identified during step 1 were completed by the end of step 2, we should be able to start step 3 without a lull in development.

The deliverables will be (a) a general-purpose library for “open” contracts in Plutus, (b) a rewrite of the previous contract to use the library, (c) support for such open contracts in *Glow*.

4.3. Step 4: A General Library for Open Contracts in the Node

The fourth and last step will be to develop a general-purpose library for malleable transactions in the node.

Assuming that any required changes to the node APIs identified during step 2 were completed by the end of step 3, we should be able to start step 4 without a lull in development.

The deliverables will be general-purpose support in the node for taking advantage of malleable transactions from step 3.

5. Bibliography

“Flashbots Transparency Report — February 2021”, thegostep
https://medium.com/flashbots/flashbots-transparency-report-february-2021-8ac45b467d0a

“Ethereum is a Dark Forest”, Dan Robinson and Georgios Konstantopoulos, August 2020,
https://medium.com/@danrobinson/ethereum-is-a-dark-forest-ecc5f0505dff

“MEV auctions considered harmful”, Ed Felten, May 2020,
https://medium.com/offchainlabs/mev-auctions-considered-harmful-fa72f61a40ea

“Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges”, Daian *et al.*, April 2019, https://arxiv.org/pdf/1904.05234.pdf

“Why Developing for the Blockchain is Hard — Part 1: Posting Transactions”, François-René Rideau, December 2018,
https://hackernoon.com/why-developing-for-the-blockchain-is-hard-part-1-posting-transactions-dde21c025c65

“How the winner got Fomo3D prize — A Detailed Explanation”, SECBIT Labs, August 2018,
https://medium.com/coinmonks/how-the-winner-got-fomo3d-prize-a-detailed-explanation-b30a69b7813f

“Chimeric Ledgers: Translating and Unifying UTXO-based and Account-based Cryptocurrencies”, Joachim Zahnentferner, March 2018,
https://iohk.io/en/research/library/papers/chimeric-ledgerstranslating-and-unifying-utxo-based-and-account-based-cryptocurrencies/