Information Theoretic Taxonomy

Notes from @cwgoes presentation at AMS 22 titled DKG/TPKE for fair ordering and a proposal to front-run cross domain MEV

Research Question:

Which kinds of MEV are possible, and which kinds can be mitigated, in principle, from an information-theoretic perspective? Developing a general taxonomy. If you want to develop a taxonomy that is general, you have to reason about it on the basis of information theory. What are the agents when are they making decisions, what are the different events where different agents have different information. If you derive a kind of structure in that framework, you can be pretty confident that it's not dependent on some specific properties of some particular system.

A very simple information theoretic taxonomy of MEV

- State of the blockchain advances in local time
- State of the world advances in real

time

- Two classes of actors:
- Transaction authors (users)
- Transaction orderers/groupers (proposers, builders, sequencers, etc.)
- Transaction authors (users)
- Transaction orderers/groupers (proposers, builders, sequencers, etc.)

We have two information systems. One of those systems is the ledger (blockchain) that advances in this quantized time stamp, could be a DAG as well. Then we have the World, which is everything else. The world includes other things like things happening in nature which may inform hedge fund trading decisions. Those are also part of the world in a general sense.

The world advances in "real" time. Then broadly we have two classes of actors. Transaction authors who are crafting a tx

which may do something they care about related to the blockchain state. They may take that state as input or prior state as input, or may depend on the state at the time the tx

is executed.

Then there are transaction orderers or groupers, these might be split into different roles. However, someone still has to make ordering decisions, or you do everything in batches and there is no order in the batches but still order across the blocks. Variously these orderers or groupers are called proposers, now in a system like Tendermint the block proposer chooses the ordering of transactions. In the Flashbots style architecture these roles are more split like in the PBS architecture. Different roles may or may not be performed by a real world entity.

Two events in question - Event of transaction authorship

- Author has access to blockchain state at time b_t
- Author has access to world state at time w_t

There is a difference here in events. The first event is transaction authorship, where the author of the transaction has access to the states at which they can read at that time. So they have access to the blockchain state, this particular block they can currently see, maybe they are using a light client, so they can see which blocks have been finalized. They also have access to the world state at w/e the current time in the world is.

Second Event - Event of tx

ordering and block creation

- Orderer has access to more information about blockchain state
- Perhaps updated blockchain state b_t+k
- Set of possible transactions to include
- Orderer has access to more information about world state

- i.e. the cross-domain MEV problem
- MEV is the information asymmetry between these two actors/events

The second event is this event of tx

ordering and block creation, which may be split into different sub events. Whoever is doing the ordering if there is ordering, even just choosing which block to put a tx

in a batch system, has access to more information about blockchain state. Maybe some blocks have passed since the tx

has been authored, that is some information they have. But they definitely have a set of possible transactions to include and possible orders for those transactions.

The orderer also has access to more information about the world state because time has passed. This is the cross-domain MEV problem writ-large. Then you can think of MEV as the information asymmetry between those choices. Between the user who is choosing to author the transaction the first time and the decision maker or multiple possible decision makers who are deciding to order transactions and put them into blocks at the second time.

If you try to diagram this in a simple fashion, where arrows are information flow. SO someone in blue is choosing to author a tx

. They have information about the world at time t = i

, and they have information about block 1 on the ledger. That transaction along with others and more information about the state of the world is available to the blob of either singular or many entities who are sequencing transactions and creating blocks. Some blocks may or may not have passed on the ledger in this particular diagram. You could also tie transactions to specific blocks, and maybe there are just more transactions and more information about the world.

Information Asymmetry

- Can we reduce the information asymmetry about the blockchain state?
- Yes, with cryptography (ordering decisions made without information about tx

contents)

- Can we reduce the information asymmetry about the world state?
- No. Only by reducing latency from event (1) to event (2)
- We can also craft algorithms to combine transactions within a quantized time period (batching)

From this taxonomy, we can characterize what are the different aspects of MEV that we could aim to reduce and how could we go about doing it. One question that might arise is can we reduce the information asymmetry about the blockchain state? There are ways we could reduce this by requiring ordering decisions to be made without information about tx

contents. Our approach falls into this category, with Threshold decryption.

Can we reduce the information asymmetry about the world state? Much harder to answer. The answer is no, because time has passed. If there is more information about the world accessible to the person who is making the ordering decision simply because time has past, we can't encrypt the world unless we pull the world into our encryption scheme. We can't do anything else, except we can try to reduce the latency. The real latency of this does matter in terms of number of seconds if the kind of MEV we are worried about is changes in the world state.

Using cryptography to reduce asymmetry (a)

- Proof-of-stake with BFT finality
- Threshold key (e.g. same shares as P-o-S validator set)
- Order of operations
- All transactions are encrypted to the threshold key
- Proposer commits to ordering
- Block finalized
- Transactions are threshold decrypted then executed

How do we use cryptography to reduce this first asymmetry (a)?

There is one component of a threshold decryption setup and another component of the settlement setup and the settlement component is pretty orthogonal, what we just care here about is the ordering component. But we happen to have this vertically integrated system. The system we've built is a DKG and threshold decryption system setup to work with Tendermint. Tendermint has BFT consensus, it is usually used in combination with a P-o-S system with regular finality 2/3 thresholds. We generate in our system (Ferveo) a threshold key which has the same share distribution as P-o-S which allows us to do this guaranteed decryption, which is critical. If you use TE and encrypt all or some of your transactions but don't necessarily have the ability to decrypt them, then someone can withhold data. If somehow there is a misalignment between how you finalize the order and how you decrypt the transactions, where it is possible to finalize an order but then not decrypt some of the transactions, you don't want that. For that reason, we integrate these systems very tightly.

In the Tendermint architecture, votes are broadcast around, and we include in those votes the decryption shares such that when you have 2/3 voted which is a commitment to a block, you also have 2/3 decryption shares which is precisely what you need to decrypt the transactions. Either you atomically finalize a block and can decrypt the transactions or not. This is like poor man's witness encryption because we don't have Witness Encryption yet, but we can fake something similar to the model. Except that, of course, 2/3 of the validators could collude and decrypt the information out of band.

A basic flow order to explain from a user's perspective. When you author a tx

in this system, you write w/e your tx

is and encrypt to the threshold key. The proposer only sees the encrypted tx

they have to commit to an ordering of encrypted txs. That block is finalized with that order finalized. Then txs are threshold decrypted and executed. This is not a scheme for transaction privacy. You could separately make parts of transactions private in another way, and you could re-encrypt them in this scheme. This scheme does not provide any LT privacy, only temporary privacy.

- Change in information asymmetry
- Proposer knows no more about blockchain state than tx

author (except gas limits)

This helps with the first information asymmetry and now the proposer isn't able to search this computational space of which transactions can be combined in which ways and make the state insert their own transactions, they don't know anything more about it than the tx

author, except gas limits.

The problem: block computation limits

- Transaction gas limits leak information
- Can choose granularity, tradeoff between packing efficiency/info asymmetry
- ZKPs do not help directly
- Only way to avoid: all transactions have identical (encrypted) sizes and execution costs
- (still much better than status quo)

There is one slight problem here. We have limits on what we can pack into blocks. Those limits may be based on different resources. They can be based simply on gas, they can be multidimensional (md1559), either way as long as you have limits on what you can pack into a block then you need to give the person who is putting things in block information about what those limits are otherwise they will not be able to ensure the block has less than w/e the limit is.

These limits leak information about the contents of transactions. If transactions are doing arbitrary programmable execution, then if you identify a specific AMM transaction always costs 120,362 gas even if that transaction is TE, you have to put the limits in plain text. That gas amount will pretty clearly identify to someone what the transaction is doing. It's more statistical information to coordinate, and there is some potential MEV there. However, you can give up some accuracy in block packing to get better privacy and less MEV in a sense by quantizing gas.

In Ethereum or most systems right now gas is moderately quantized in the sense we have 120,000 gas 120,001 gas. If you use an additional EVM opcode in a transaction, it will change the gas costs. But if we make gas less granular by rounding everything up to the nearest multiple of some constant we choose, we can round everything up to the nearest 100th then we are revealing less information about the contents of transactions and there is less opportunity for MEV.

ZKPs don't directly help with this problem. If you are doing private transactions where you are pinning to a previous state, then you are not leaking information in this way. But if you just use ZKPs to prove something about the gas consumption, you still have to reveal what the gas consumption is.

How do you avoid this? You give up the generality here. You have to make all transactions have identical encrypted sizes and execution costs. Even if you are just doing DA sequencing, and you're not executing transactions on chain if you are ordering them on chain, and they have different sizes, and you use this TD scheme, it still reveals something when the ordered is picking which ones to include. Still, even with gas limits leaking some information, this seems like it's probably better than the status quo in terms of MEV.

What do we do about cross-domain MEV?

- Cross-domain is like information in the world (prior diagram)
- Quantize time (artificial latency) by shared clock
- Synchronize encryption-ordering-decrypttion
- What doesn't this do: cross-domain messages
- Pre-encrypt if you know ahead of time
- Chain encrypt is you have state dependence (hard)

Cross domain MEV we are talking about different domains which are different ledgers. If we don't coordinate anything between those domains, then it's just like the other domains are part of the world. If a lot of relevant information is happening in other domains, that would cause there to be MEV opportunities like price changes on other exchanges on other ledgers that will create a lot of MEV here based on the real-time which passes that can be exploited.

But if we coordinate between the domains, we have more opportunities. In particular, if we are using this kind of TD scheme, we can coordinate on not necessarily what exactly who is doing the encryption or decryption, but we can coordinate when we do it. If we have this cross domain system of three ledgers. Let's say we have transactions being created, then we have some kind of shared deadline and some type of interval between a deadline and the shared decryption. So all the ledgers have to agree. One way of agreeing is using real world time-stamps and hope that those are measured correctly. You could send messages b/w the ledgers, but that is more complicated and introduces liveness assumptions. Maybe in practice a reasonable way to coordinate is just by using timestamps. Then, if you require that to be included, all the transactions must be created by this particular deadline. They are encrypted, all the ledgers commit to ordering them in a particular fashion, then after some slight interval they are all decrypted and executed. Then you get similar properties, at least in the sense of transactions on one ledger not immediately creating MEV opportunities on another ledger.

Conclusion I

- For the basic approach, we don't need to agree on the specific encryption scheme or keyset or DA layer or execution layer- we just need to agree on the clock!
- Cross-domain message passing MEV is hard, but this is a concurrency problem, not a security problem (unless your bridge breaks)
- Rollups will have some architectural challenges

The nice thing about this approach is we don't need to agree on the specific encryption scheme, or key set or DA layer or EL layer, we just need to agree on the fact that we are doing threshold decryption, and we need to agree on the clock.

What this does not solve is cross-domain message passing MEV; i.e. cross-chain DEXs might not work. Which is that if you are making state changes on one chain and those cause messages to be issued to another chain and those messages reveal information then you can threshold encrypt them all you want as soon as you decrypt the messages which will be sent to another chain that creates the MEV opportunity and there will be some competition to fill it in the next round.

One interesting corollary that at least I've come to is that this problem has nothing to do with security, unless your bridges fail. That's not really an MEV problem unless MEV is making it more profitable to bribe people who are running your bridges. But mostly this is a concurrency problem. And if you have a single chain with different rollups on a DA layer, you have the same problem with cross rollup messaging, and you will face the same design constraints.

Conclusion part II

- Nothing to be done about world-MEV, but physical change is slow
- In the limit case (no CEX), this is a non-problem
- In the non-compliant cross-domain case, real-world latency matters

It doesn't seem like there is anything to be done about world-MEV. Luckily, physical change is slow, if you are looking at processes occurring in nature and something from one second to the next changes in ways that radically change the price of the asset, that may be concerning or create MEV opportunities, but it's not clear what those changes would be.

Another corollary is that real-world latency matters, like the actual latency from when you submit the transaction to when the decision about ordering is made really matters. If there is more latency, there is more MEV.