What EVEN is commitchain?

One approach for blockchain scalability derives from a simple question:

> *Why do Alice and Bob send every transaction to a global network for settlement?*

What if both parties, Alice and Bob, could transact locally amongst themselves and only send a single, and final update, to the global network.

This is the thought-process and idea behind layer-2 scalability. It simply reduces the number of transactions processed by the global network.

Layer-2 scalability has been with us all along.

At a high level, most layer-2 systems (except channels) have common components:

- **Locking funds (deposit).** A user has to lock funds into the service via the layer-1 blockchain.
- **Off-chain database.** A list of accounts and their respective balances.
- **Sequencer**. One, or more entities, who are responsible for collecting the layer-2 transactions and deciding the order of execution.
- **Off-chain onboarding.** A user can receive coins on the service without any prior interaction with a blockchain.

It is not immediately obvious, but the most popular and widely used example of a layer-2 system are cryptocurrency exchanges. They have been in operation for over 10 years, there are several popular services, and they all effectively extend the functionality of cryptocurrencies while keeping most transactions off-chain. However, they are custodial systems and this brings us to the final properties of a non-custodial layer-2 protocol:

- **Publicly verifiable.** Any participant can verify the offchain database's correctness,
- **Layer-1 security.** The protocol's integrity and security is derived from the layer-1 blockchain.

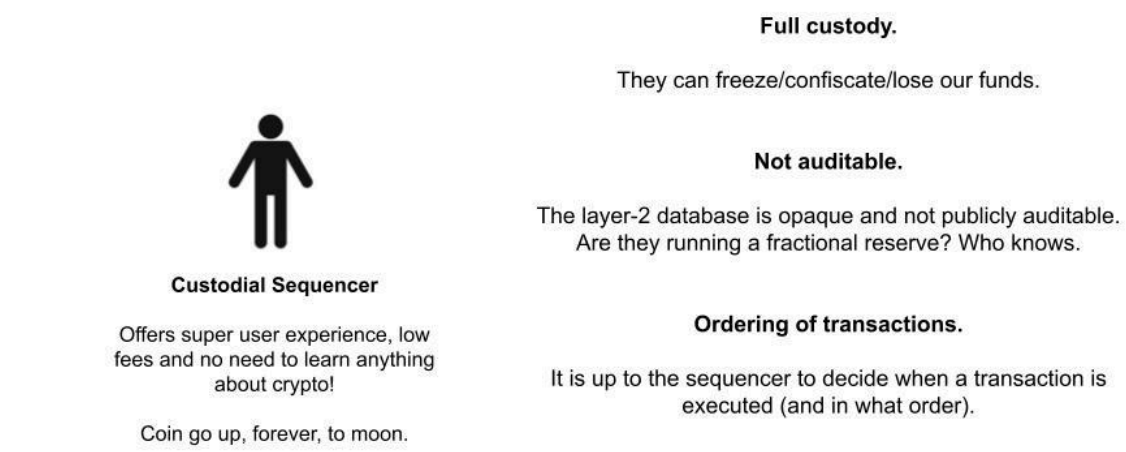


Figure 1: Historically, layer-2 has had fully trusted and custodial sequencers.

Cryptocurrency exchanges are fully custodial and do not resemble the layer-1 blockchain, in any manner, in terms of security. Exchange operators can freeze and confiscate our funds, and even worse, they have already lost billions of dollars worth of funds over the years. Ultimately the user must give up control of their funds to use the service.

On the other hand, the off-chain database of account balances is not auditable or publicly accessible. It is impossible to verify whether a cryptocurrency exchange is running a fractional reserve (or in other words, if they have lost the user's funds). While proof of reserves protocols have been proposed to help alleviate the fractional reserve issue, the protocols are very cumbersome to implement and exchanges have little incentive to pursue them.

Today, we blindly trust exchanges with our funds. This brings us to the final design goal for a layer-2 system:

Can we bypass transacting on the layer-1 blockchain
while still allowing users to maintain self-custody of their funds?

Yes, it is possible. Layer-2 protocols leverage on-chain smart contracts to hold custody of the user's funds and to constrain how the third party service can interfere (or manipulate) a user's transaction. If successful, the intermediary should become a sequencer who is only trusted to sequence and finalise transactions (i.e, a sequencer). However, the design and implementation of such a system is non-trivial. It brings a host of challenges that we will explore throughout this article including whether we need to review the sequencer's reputation, how the off-chain data is published for public review, the process for finalising a layer-2 transaction, and of course, the extent to which we can truly bypass the layer-1 blockchain when transacting.

Let’s explore rollups, and more generically, commitchains, from first-principles.

Prominent voices have proposed a definition that is widely used to inform non-experts what it means to be a layer-2 protocol:

> *A system is only considered layer-2 if the protocol’s integrity and security is derived from the parent blockchain (Layer-1).*

The motivation is to provide clarity and distinguish that protocols, like sidechains, are not layer-2 protocols.

However, it is still an ambiguous definition.

It does not break down what it really means to derive integrity or security from the layer-1 blockchain. Our task is to unravel the ambiguity and as we will see, it is impossible to fully resemble the same security model as the layer-1 blockchain. But it is possible to get pretty damn close and it is this *closeness* that becomes the distinguishing features of different layer-2 protocols.

We will explore the following:

- What is a commitchain and how does it work?
- What are the protocol assumptions and the wider environment in which the commitchain is deployed?
- Who is the adversary and what is their power?
- What are the security goals that must be satisfied before a commitchain is “secure”?

Afterwards, we evaluate the mechanisms and solutions used by layer-2 protocols to get “close” to the security of the layer-1 blockchain.

Overview of layer-2 protocols

Generally speaking, the goal is to move transaction processing from the layer-1 blockchain to the layer-2 blockchain. If we are successful, then only the local parties involved in the transaction need to process it and not the global peer-to-peer network.

Transaction processing can be broken into:

- **Data.** The bytes of the transaction. It includes the target contract address to execute, the function name and its inputs. It can also include the expected output of the transaction.
- **Computation.** Replicating the transaction execution to verify the function and input corresponds to the expected output.
- **State updates.** The smart contract state, which includes all variables and its bytecode, is updated after processing the transaction.

The best case scenario is to keep the data, computation and state updates in the layer-2 system, and this is what we call an off-chain protocol. However, over the years, we have witnessed unique challenges at keeping all three aspects off-chain. As a result the most popular approach, the rollups, only keep computation and state updates off-chain. We still call this a layer-2 protocol as it is a mix of local and global transaction processing.

There are two flavours of layer-2 protocols:

- **State Channels.** A small set of parties lock funds and continuously replace transactions amongst themselves. It requires all N parties to agree on every state transition and the layer-1 blockchain is only consulted if there is a dispute.
- **Commitchains.** A chain of checkpoints, proposed by a sequencer to the layer-1 smart contract, that dictates the order of execution for all layer-2 transactions.

Our article is only focused on commitchains. In a future article, we will discuss how state channels alongside other approaches can solve the problem of routing value across multiple ledgers (and commitchains).

What is a commitchain?

At a high level, we need to first design a new blockchain network.

A blockchain network. A ledger that records the state of all accounts. It can have its own smart contract environment with a virtual machine and programming language. Users still have a private-public key pair to authorise transactions that will update the ledger's database once the transaction is finalised in the blockchain. And of course, we have sequencers (block producers), who will come to agreement in regards to the longest chain and the transactions it includes.

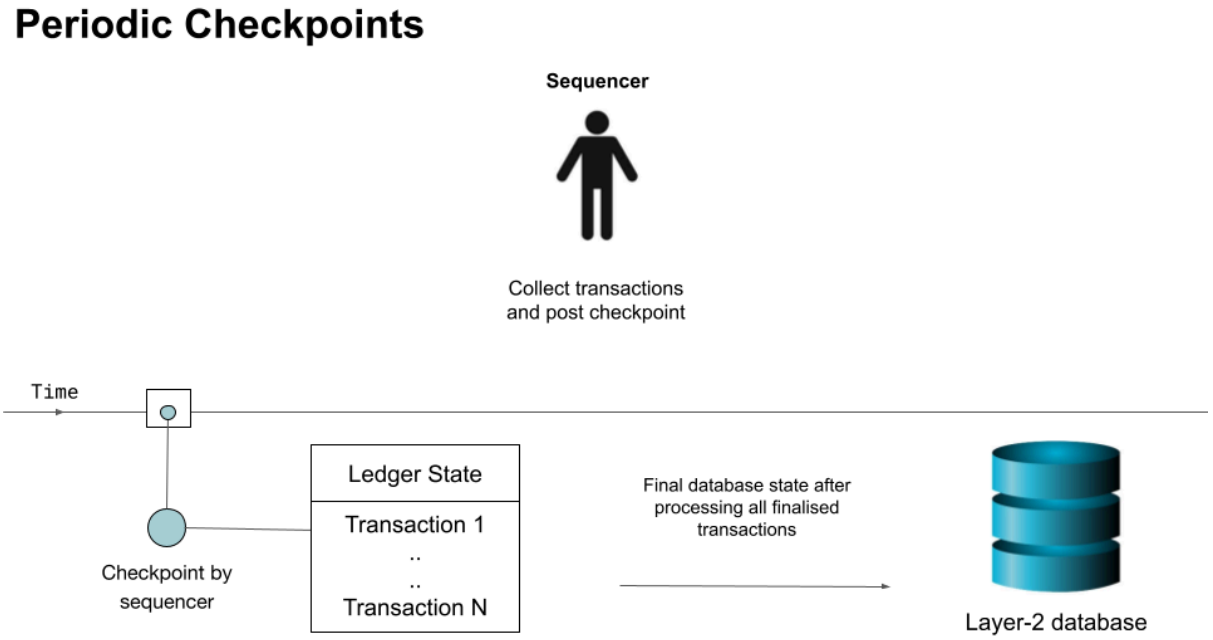


Figure 2: At a high level, the sequencer posts checkpoints that can be processed to re-compute the layer-2 database.

Unlike a new blockchain network, a commitchain, as the name suggests, is a chain of commitments that are posted to the layer-1 blockchain:

- **Checkpoint (commitment).** A cryptographic commitment to a list of transactions alongside the new ledger state.

A commitchain is illustrated in Figure 2 and there are two agents who are responsible for managing the commitchain:

- **Sequencer.** Responsible for collecting and ordering layer-2 transactions, and for periodically posting a checkpoint to the layer-1 blockchain.
- **Layer-1 smart contract.** Responsible for defining and enforcing the validity rules for the commitchain.

It becomes a prover-verifier model, where the sequencer (prover) wants to convince the layer-1 smart contract (verifier) that the commitchain is correct. The layer-1 smart contract will only accept a new checkpoint as valid if:

transition(list_of_transactions, previous_ledger_state) === new_ledger_state

Put simply, every state transition for the layer-2 database must be verified and enforced by this layer-1 smart contract. However, the layer-1 smart contract is computationally limited and it cannot re-execute all transactions itself. Thus, a third party is necessary to assist the layer-1 smart contract to verify the commitchains correctness.

How the third party is constructed (mathematically as a zero knowledge proof, or a fraud proof challenger) is often the most significant topic amongst rollup providers. It has implications on the capability and functionality for the layer-2 protocol. We'll dive into it shortly.

As a final note, it is the layer-1 smart contract and not the sequencer, who holds the user's funds and only allows the funds to be spent if the user has authorised it.

Understanding and comparing rollups

Security model for layer-2 protocols

In this section, we'll cover the protocol assumptions, the wider environment and the adversarial model for commitchains. Once we have the setup outlined, we can cover the security goals for a commitchains, or in other words, what it really means for a commitchain to be secure.

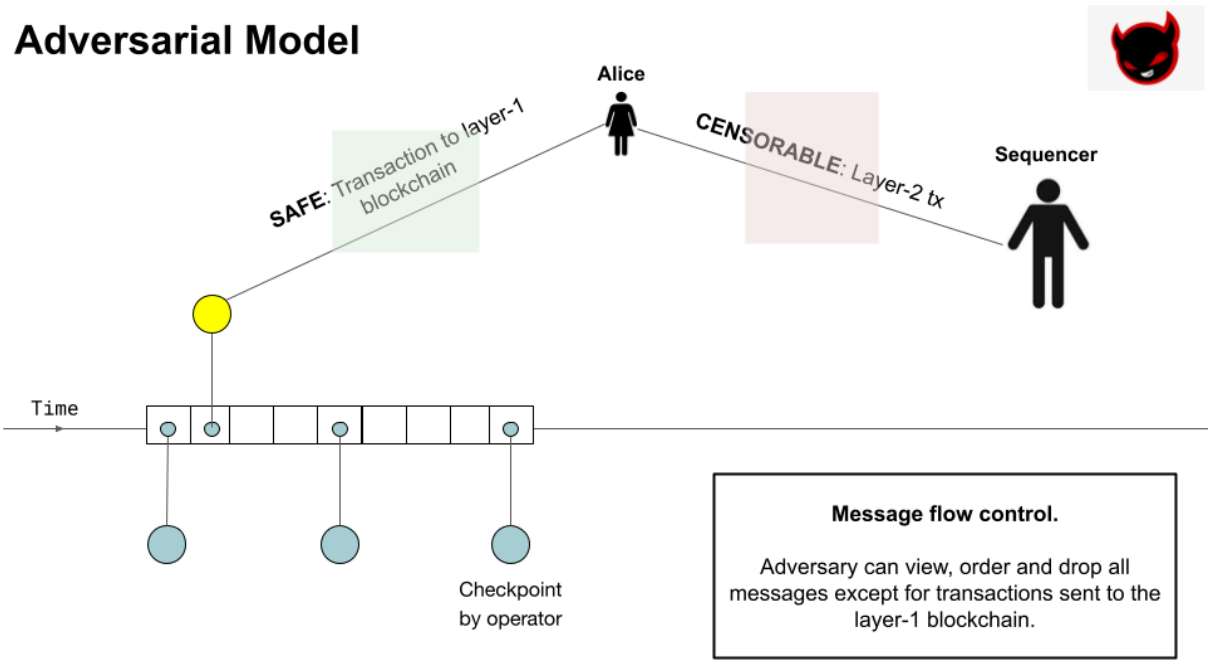
Protocol assumptions

To summarise, we assume the following about the layer-1 blockchain:

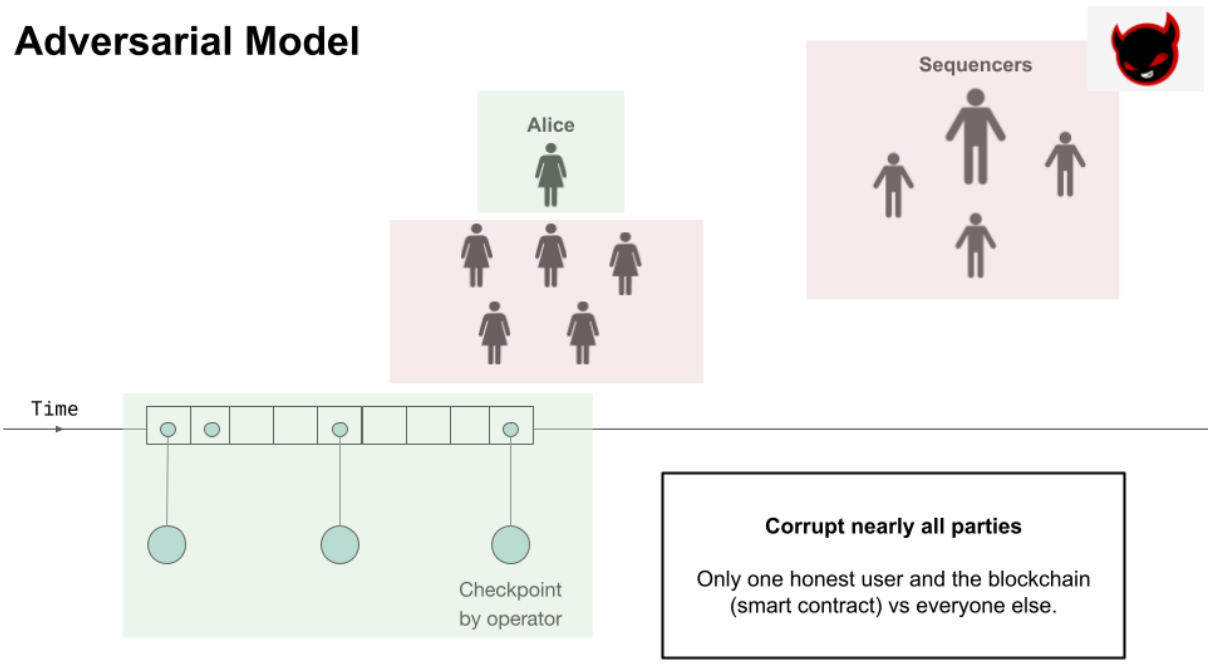
- **Neutral infrastructure.** The layer-1 block producers are independent and financially rational.
- **Consistent view.** All users have access to the parent blockchain and its historical transaction data.
- **Eventual delivery.** The parent blockchain will accept a recently published transaction within N blocks if it pays an appropriate network fee.
- **Smart contracts.** A smart contract is a trusted third party with public state and its underlying blockchain is immutable & cannot be compromised.
- **Limited computation.** The parent blockchain (and smart contracts) has considerably less computational resources than the commitchain network.

Adversarial threat model

A threat model lets us define the power and strength of an attacker who may want to break our system. The Dolev-Yao model is typically used for security protocols. It is a powerful attacker who can read and control the flow of all messages. The only restriction placed on the attacker is that they cannot compromise well-known cryptographic protocols (and of course, protocol assumptions). Layer-2 protocols should assume a Dolev-Yao adversary:



Message flow control. The adversary can control the order of all messages except for messages sent to the parent blockchain (e.g. eventual delivery protocol assumption) and they can selectively drop messages in the layer-2 protocol (e.g. censorship). Note, if the network has a gossip protocol, then it weakens the adversary such that they can only control what transactions are included in their blocks but not what transactions are propagated across the layer-2 network.



Corrupt nearly all parties. The adversary can corrupt all sequencers (e.g., there is effectively only one sequencer) and N-1 users (e.g. all users except the honest user). We assume the adversary cannot interfere with the parent blockchain's execution (smart contract protocol assumption). As a result, there are two honest users, the user themselves and the parent blockchain. Note, there are some layer-2 protocols that will weaken this threat model by assuming the user can become a sequencer and thus there is at least 1 honest sequencer at some point in time.

Well-known security properties for layer-2 protocols

To recap, we assume the parent blockchain will reliably accept transactions within X blocks and that the adversary can corrupt all parties except for the layer-1 smart contract and an honest verifier.

With this backdrop in mind, it allows us to provide a list of security properties that a layer-2 protocol should satisfy in order for the system as a whole to be considered secure:

- **Data availability.** Users can always access the layer-2 system's historical transaction data in order to re-compute the entire off-chain database,
- **State transition integrity.** The parent-chain will enforce the integrity (correctness) of a roll-up block. No transaction can perform any invalid state transitions,
- **Withdrawal integrity.** Given the rollup data, the rollup checkpoints and the parent blockchain, a user can withdraw their coins without the sequencer's cooperation.

If all properties are satisfied, then it ensures a user can re-compute the entire off-chain database (or at least, compute its latest state) and independently verify its integrity and correctness. And in the worst-case scenario (e.g. layer-2 system goes offline), then the user can safely withdraw their funds in a timely manner.

As we will soon see, how a layer-2 protocol satisfies these properties will result in practical tradeoffs that may not withstand an all-powerful adversary and instead require additional protocol assumptions for security.

What about the less well-known security issues that arise?

While the sequencer cannot tamper with the transaction content or prevent a user withdrawing their funds, there are other areas of concern:

Maximal extractable value. The sequencer can control the ordering of transactions. They can decide to drop (censor) your transaction, or just like on Ethereum, they can attempt to extract value by sandwiching, front-running or back-running your transaction. There is an on-going debate whether extractable value is an outright attack on the system or if it is a potential business model for layer-2 protocols. Thankfully, it is possible to constrain extractable value to a certain extent via smart contracts (i.e., think of uniswap slippage controls).

Fast path or path. Assuming the sequencer is well-behaved, then the user can enjoy the fast path for transaction confirmation. The user can send the sequencer a layer-2 transaction and safely assume it will be eventually mined and executed as expected. However, if the sequencer does not have a good reputation or if the layer-2 protocol does not guarantee which sequencer can send the next block of transactions, then the user must wait until the layer-2 transaction is finalised and they cannot simply accept an acknowledgement from the sequencer. In the worst-case, the user may need to send the layer-2 transaction to the layer-1 smart contract directly (if supported). Thus, it is important to consider how sequencers are appointed and the consensus protocol (e.g. how to reach agreement on the transaction execution) when evaluating the layer-2 protocol's user experience.

Mass exit problem. All user deposits are locked into a layer-1 smart contract and it can be described as a 'pot of funds'. Each user has a potential claim to some funds in the pot according to their balance in the layer-2 system. It is possible that due to a bug or an unforeseen attack, the system may become a fractional reserve due to a loss of funds. If so, then the pot of funds is no longer sufficient to honour all user withdrawals. This becomes a `mass exit problem` as every user has an incentive to exit the system before other users as it is the last remaining users who will bear the loss. Of course, there are other reasons why a mass-exit can be triggered, but it typically boils down to loss of funds, or loss of access to withdrawing funds

Security properties definitions & explanations

In the following, we will explore the defined security properties alongside the issues that may impact the security of a rollup. For each issue, we iterate on a list of solutions that can help alleviate it.

Sequencer Profile

As mentioned previously, sequencers offer a fast-path for transaction inclusion in the rollup chain. Our sequencer profile focuses on how to appoint a sequencer and what rate-limits who can become a sequencer, the consensus protocol that governs how to progress the rollup chain and finally whether the sequencer can provide any assurance to the user about the final state of their transaction while it is still off-chain and pending. pending off-chain.

Sequencer Appointment Protocol

In most rollups, a fast-path for transaction inclusion is offered by sequencers. They are responsible for accepting a user's transaction and responding with an acknowledgement that the transaction will eventually be finalised. We call this the "fast-path" as sequencers are crucial to the optimistic operation of a rollup. As such, the identity and reputation of a sequencer remains important, and there must be an appointment protocol to rate-limit, in as-close-to permissionless manner, who can offer the fast-path to users.

Central Authority. The sequencer is pre-determined by the team who set up the system and no one else can participate.

Vote. A DAO, or some other voting mechanism, can vote in a set of sequencers to run the network.

Stake. Any user can become a sequencer if they are willing to stake X value in the rollup contract.

Consensus Protocol.

A consensus protocol allows one or more agents to reach agreement on a single decision. In the case of rollups, the sequencers may need to agree upon the batch of transactions to order and/or the final execution of the transactions.

Central Authority. A single sequencer is responsible for publishing

Round robin. The bridge contract provides a time slot for each sequencer to submit a new batch of transactions and checkpoints. If the sequencer misses their allocated time slot, it will move onto the next sequencer. TOOD: Binance Smart Chain is a round-robin, although it is not a rollup.

Majority vote (PBFT-like). A set of sequencers must agree upon the next checkpoint

Off-chain assurance for layer-2 transactions

All public rollup networks will have a public peer-to-peer gossip protocol for propagating new pending transactions. The computational, storage and bandwidth requirements may far exceed the layer-1 blockchain (Ethereum) and the number of validators who can keep up in real-time may be reduced. As such, the rollups are investigating protocols for providing

guarantees to the user (a light client) about the status of their transaction and how it will be executed.

Trusted sequencer: meaning there is only one sequencer centrally maintained by the organization so it can be held accountable. They have only 1 trusted one now and want to move to decentralized fair sequencer at some point in the future.

Transaction receipt: The user receives a signed receipt from the sequencer that their transaction will be included in the next batch of transactions sent to the layer-1 blockchain. If the transaction is not included, then the user can provide evidence to the on-chain smart contract and punish the sequencers. Who signs the message depends on the consensus protocol by the rollup (e.g.a set of validators, then a super majority may be required). It does not necessarily promise a fixed position in the transaction queue.

Temporary fact-blocks: The sequencer will publish to a peer-to-peer network a new block that confirms the execution for a batch of transactions. This is considered a 'fact' as the execution is confirmed and any user can publish the fact-block to the blockchain. The blocks are considered temporary as the canonical chain and the final ordering of blocks is decided by another process at a later stage.

Fair-ordering protocols. A group of sequencers will work together to ensure that layer-2 transactions are acknowledged and the final transaction ordering that is sent to the layer-1 blockchain is "fair". Here, fair implies no single sequencer can impact the ordering of transactions for their own gain.

MEV auctions. A sequencer will publish a list of pending transactions and searchers will propose an ordering that maximises extraction of value. There is no guarantee a transaction will be included or its final position in the queue.

Finality guarantees by layer-1

This relates to the order in which the layer-1 blockchain receives the layer-2 transactions for global ordering and when the layer-1 blockchain is convinced about the final execution of a layer-2 transaction. It does not take into account any off-chain promises the sequencers have offered its users as the promises can be ignored (and potentially slashed in response).

Global ordering before execution. The layer-1 blockchain is responsible for maintaining a queue of pending layer-2 transaction data and it will enforce future checkpoints posted by sequencers that layer-2 transactions will be executed in the same order as the queue. Sometimes it is called an "inbox" smart contract.

Execution before global ordering. The layer-1 blockchain will record "facts" about a transaction's execution and how the layer-2 database should be updated. The fact may include a single transaction or a batch of transactions. However, the fact is not necessarily ordered or included in the final canonical chain. This is a separate process that occurs after the fact is posted.

Globally ordering and execution simultaneously. The sequencer will post the layer-2 database updates (transactions) alongside the checkpoint to confirm their execution. It is a single process and the ordering/execution is determined at the same time.

Data Availability

A solution to guarantee a rollup's history is publicly available. Note, depending on the solution to state transition integrity, the history does not necessarily need to include data on "why" the layer-2 database is correct and instead it should contain sufficient data to allow a user to independently re-compute the layer-2 database themselves. This is useful for the user as they can verify the database's integrity and inspect the latest finalised state for accounts (and contracts) on the network.

There are four solutions to the problem:

- **Trusted party (or committee).** An external set of parties will sign a message to vouch that they have the transaction data and the signatures are submitted to the bridge contract such that it can verify the attestment.

- **On-chain challenges for data availability.** A checkpoint has a challenge period (such as two weeks) and a user can challenge the state defender to reveal data on the blockchain. (i.e., all data is organised as leaves of a merkle tree and the user can request for "leaf 5" to be revealed).

- **Transition history is attached to a database entry.** A coin is fixated to an entry in the layer-2 database. The coin can only be withdrawn if the owner can prove that all state transitions (up to this point) are valid and they are indeed the real owner.

- **Post data to the layer-1 blockchain (Ethereum).** All data is posted to the layer-1 blockchain and the bridge contract will verify it is available before accepting a checkpoint.

State Transition Integrity

The bridge contract on the layer-1 blockchain is responsible for checking that all posted checkpoints for the rollup are valid, well-formed and it does not contain any invalid transactions. There are two approaches for solving the problem that include fraud proofs and validity proofs.

Fraud proof.

Given a new checkpoint from a state defender, Ethereum will wait for a fixed period of time before considering the checkpoint final. Any external observer has ample time to provide evidence (proof of fraud) that the checkpoint contains an invalid state transition and it should not be accepted. There are two fraud proof systems:

- **One-round fraud proof.** There is a commitment to every state transition that moves from the previously finalised checkpoint to the new asserted checkpoint. An external observer can send the pre-state, the transaction, and the post-state to the layer-1 blockchain. The layer-1 blockchain can verify there is an intermediary state commitment for the pre-state and the post-state. It will execute the transaction (given the pre-state) and it will compare the computed post-state with the checkpoint's commitment. If the post-state does not match, then the checkpoint is considered valid.

- **Multi-round fraud proof.** Given two checkpoints, the state defender and the challenger agree upon the number of instructions. They perform a binary search on the list of instructions until they identify an instruction they disagree upon (disputed). The layer-1 blockchain will execute the single instruction to determine if it is invalid or not.

TODO: I wonder if it makes sense to have a small table that compares the two fraud proofs?

Validity proof.

Ethereum will verify a proof alongside the checkpoint and it will be immediately convinced the checkpoint is valid. (i.e., it is proven beyond reasonable doubt that there are no invalid state transitions).

TODO: Is there a good way to explain the differences in the proving systems? Typically we have trusted setup, proof size, etc. But it is not that great a distinction compared to the fraud proofs.

Self-enforcing State Transitions

A user's funds must remain safe in the event the rollup network is offline or the sequencers are malicious. Ideally, the user should be able to transact to unwind their positions in some smart contracts and then to withdraw their coins from the bridge contract. This requires the bridge contract to self-enforce liveness and progress on the rollup network based on transactions submitted by the users. There are a few solutions to the problem:

- **No censorship resistance.** All transactions must be proposed by the sequencer and the user cannot self-enforce a transaction without the sequencer's cooperation.
- **On-chain transaction queue.** The bridge contract maintains a queue of pending layer-2 transactions for execution. Any user and the sequencer can submit transactions to the queue for ordering. Typically, the transactions sent by the sequencer have priority in the ordering. When the final transaction ordering is set by the bridge contract, then a state defender can submit a new checkpoint that asserts the final execution of transactions. decides the final ordering of layer-2 transactions for execution. An analogy for this approach is a message "inbox" and all users can send messages to the inbox.

- **Permissionless sequencer.** Anyone can become a sequencer (depending on the rate-limiting mechanism used) and they will eventually have an opportunity to include a transaction in the next checkpoint.

Max Exit Potential

If the security guarantees of a rollup system is broken, then it is possible for the bridge contract to no longer allow the user to redeem their full balance back onto the layer-1 blockchain. A mass-exit is when users can foresee that the security guarantee will be broken and compete amongst themselves to remove their coins before others. There are two events that can result in mass exits, potential-claim risks and data availability risks.

Potential-claim risks. In most rollups, there is not a one-to-one linkage of coins held by the bridge contract and the coins used on the rollup network. Instead, the coins on the rollup represent a potential-claim to a corresponding quantity of coins in the bridge contract. If the state transition integrity property is broken and the adversary can remove a portion of the coins held by the bridge contract, then it will no longer be possible for users to redeem their full balance. As such, the first users to exit will redeem their balance and the final users will lose their balance.

Data availability risks. Users need to access the layer-2 database in order to convince the bridge contract that they have a positive balance and their intent to withdraw it (including unwinding any positions). In most rollups, the latest state of the layer-2 database is required to interact with the bridge contract and if the data is missing then the user cannot self-enforce state transitions with the bridge contract. As a result, if it becomes clear that the data will not be available in the next epoch, then all users will rush to withdraw their coins before the data becomes unavailable.

Performance

In this section, we study the scalability of the proposed rollup protocols. We use a single metric for assessing scalability, gas used by the layer-1 blockchain, as that is the ultimate bottleneck faced by all rollups.

Data type

We consider the type of data that is sent to the layer-1 blockchain. This data can be split into two components. It may contain the “why” this transaction is valid and it will contain the “how” to update the layer-2 database.

Full transaction. A transaction that contains the user’s signature, the function and data to be executed, and miscellaneous data

Compressed transaction. A transaction that contains instructions on how to update the database, but it does not contain data that proves it is valid. For example, it may require the verifier to execute a function, but it does not contain a signature from the user to authorise it. (i.e., no signature from the user).

State diff. A database update that instructions how to modify existing state. It may not contain transaction details, it may represent the results of more than one transaction, and it will not contain evidence for why it is a valid update. For example, it may state to update Alice's balance by one coin.

As a note, a compressed transaction or state diff is mostly only possible with zero-knowledge rollups. A zkproof can vouch for why the database is valid without publishing the full details of a transaction. There is some work towards compressed transactions for optimistic rollups, but so far it is application-specific and limited.