CONSENSYS

Scaling Ethereum using Roll-ups and Sidechains

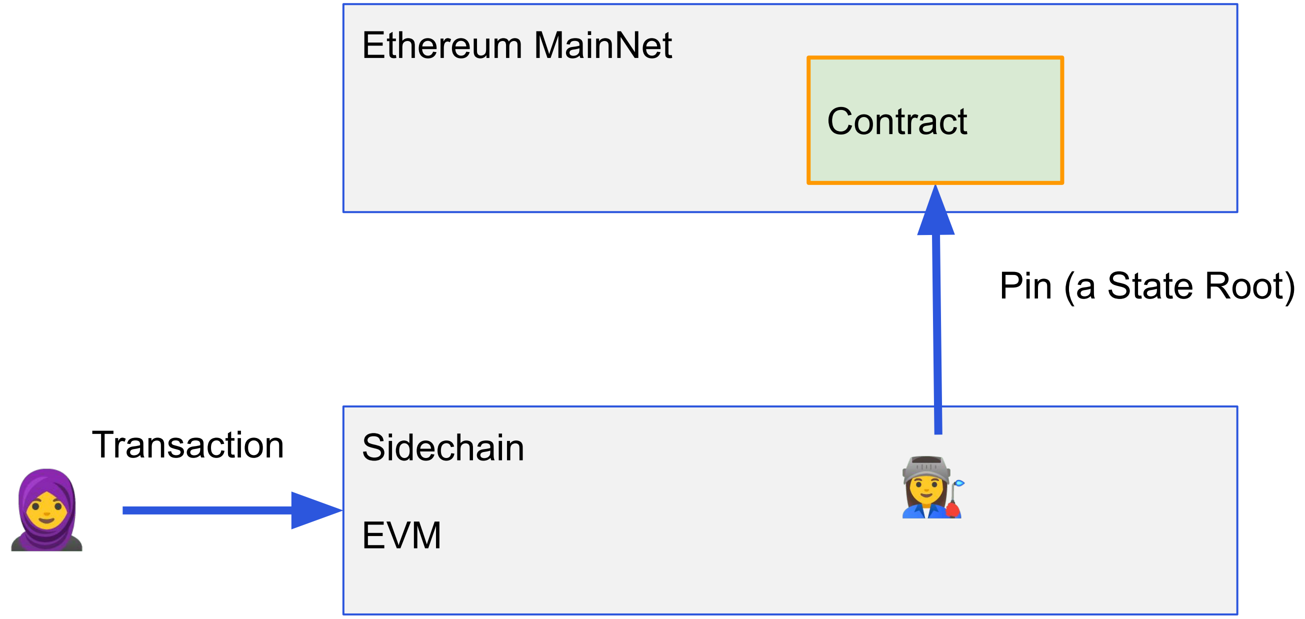
Peter Robinson

June 9, 2021

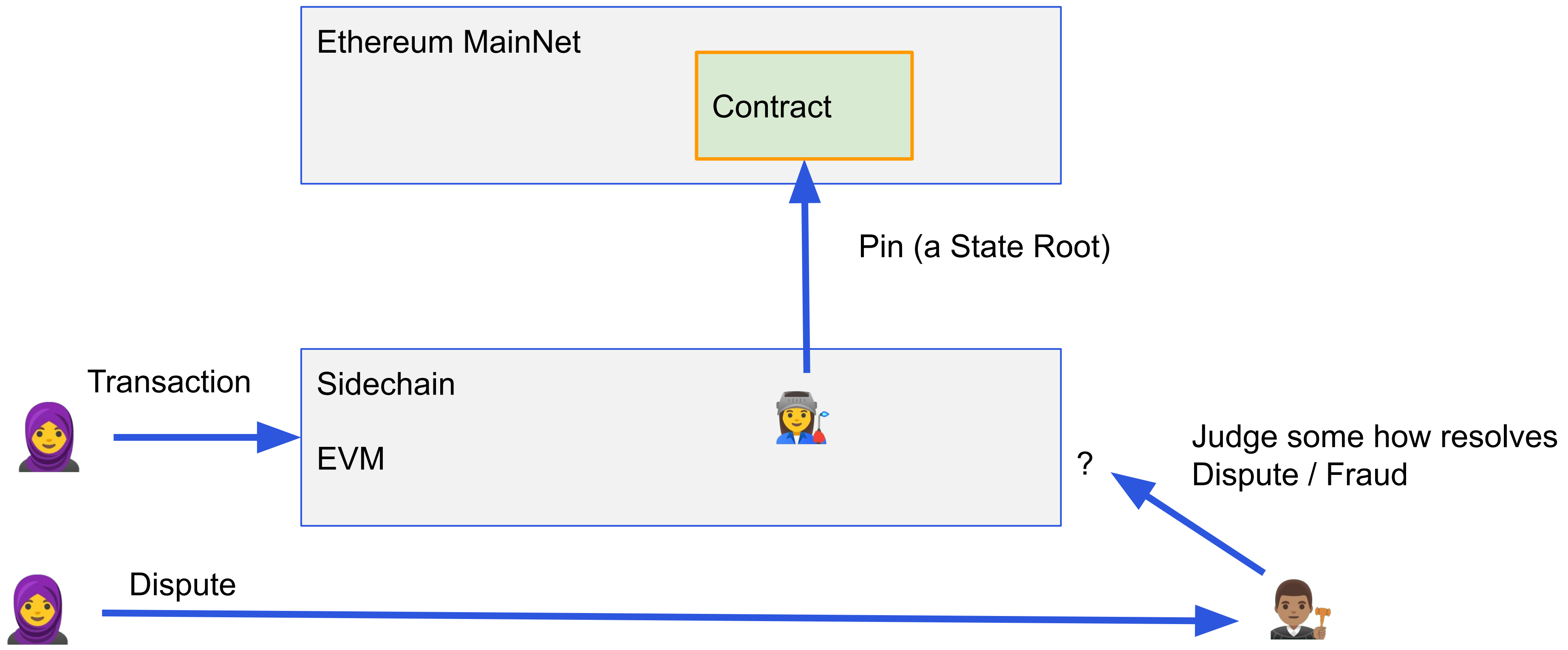
Agenda

Overview

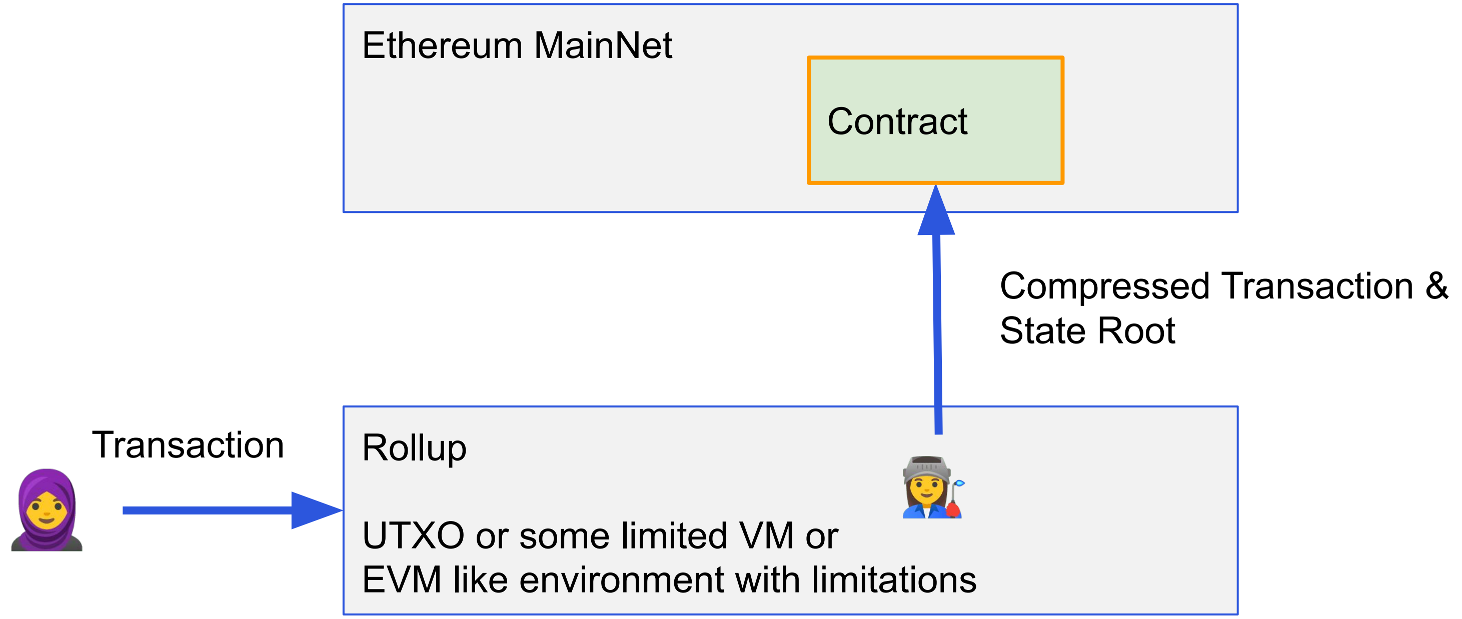
Sidechains


Ethereum MainNet
Contract
Pin (a State Root)
Transaction
Sidechain
EVM

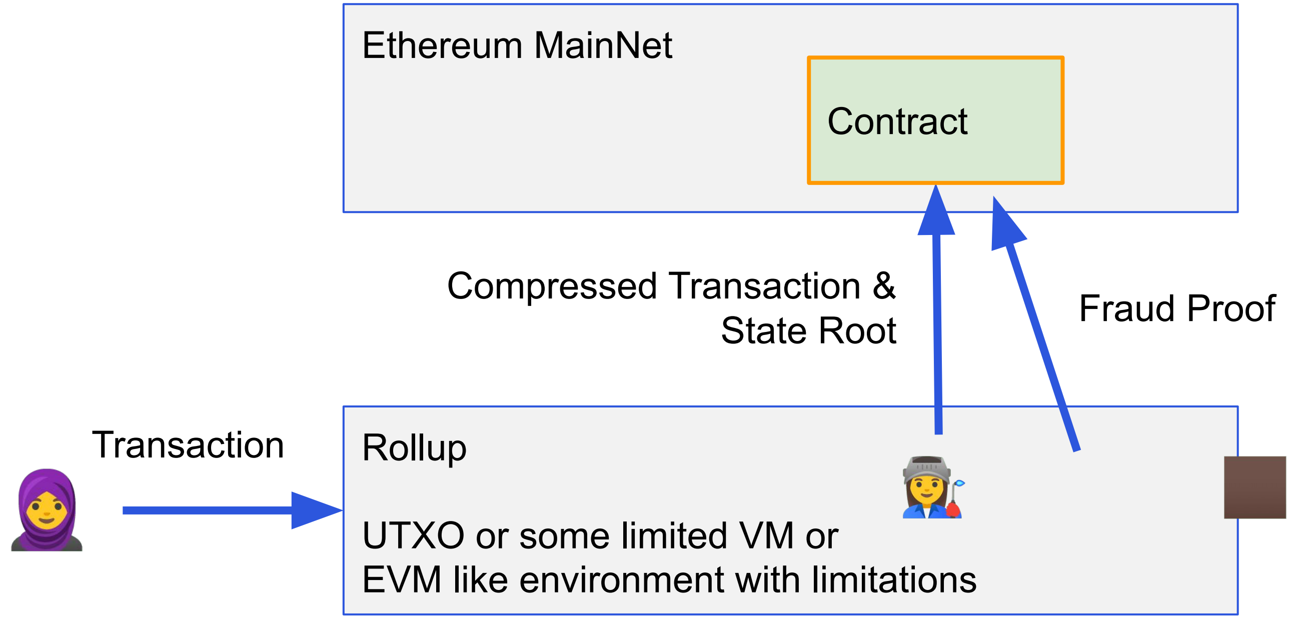
Sidechains


Ethereum MainNet
Contract
Pin (a State Root)
Transaction
Sidechain
EVM
?
Judge some how resolves Dispute / Fraud
Dispute

Optimistic Rollups

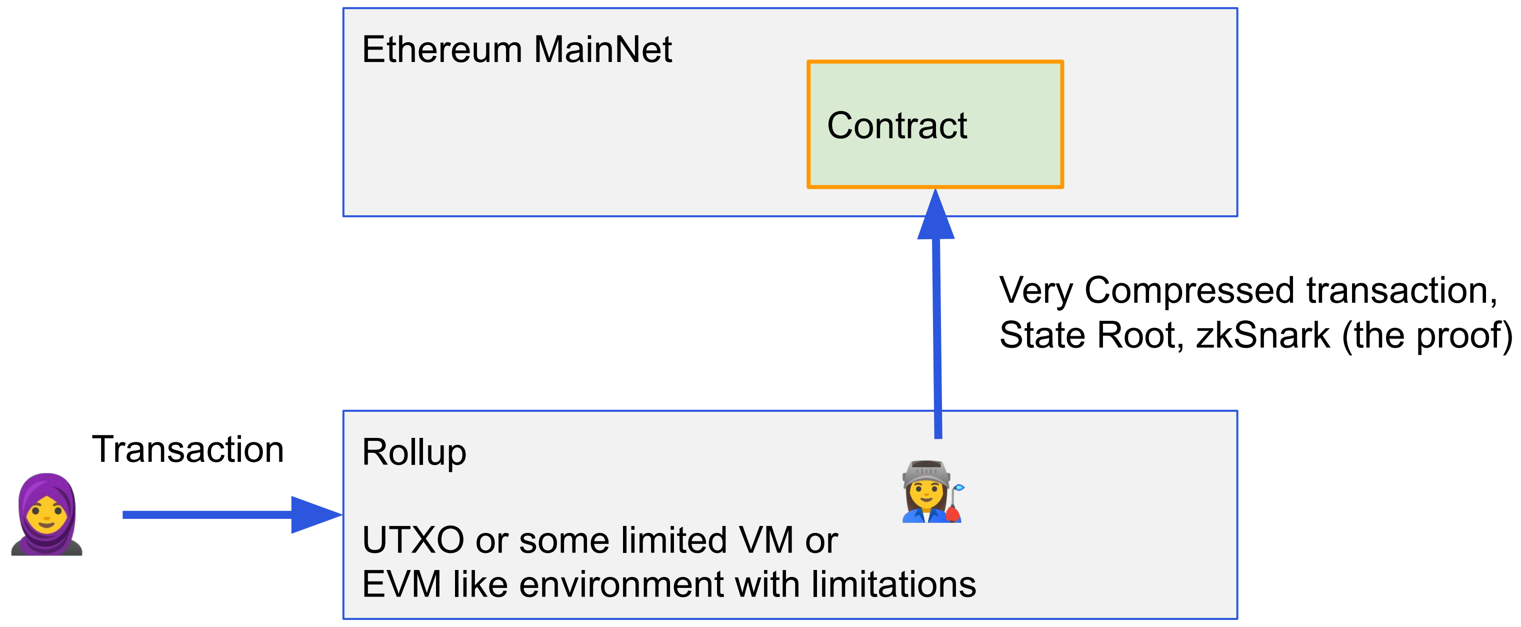

Ethereum MainNet
Contract
Compressed Transaction & State Root
Transaction
Rollup
UTXO or some limited VM or
EVM like environment with limitations

Optimistic Rollups

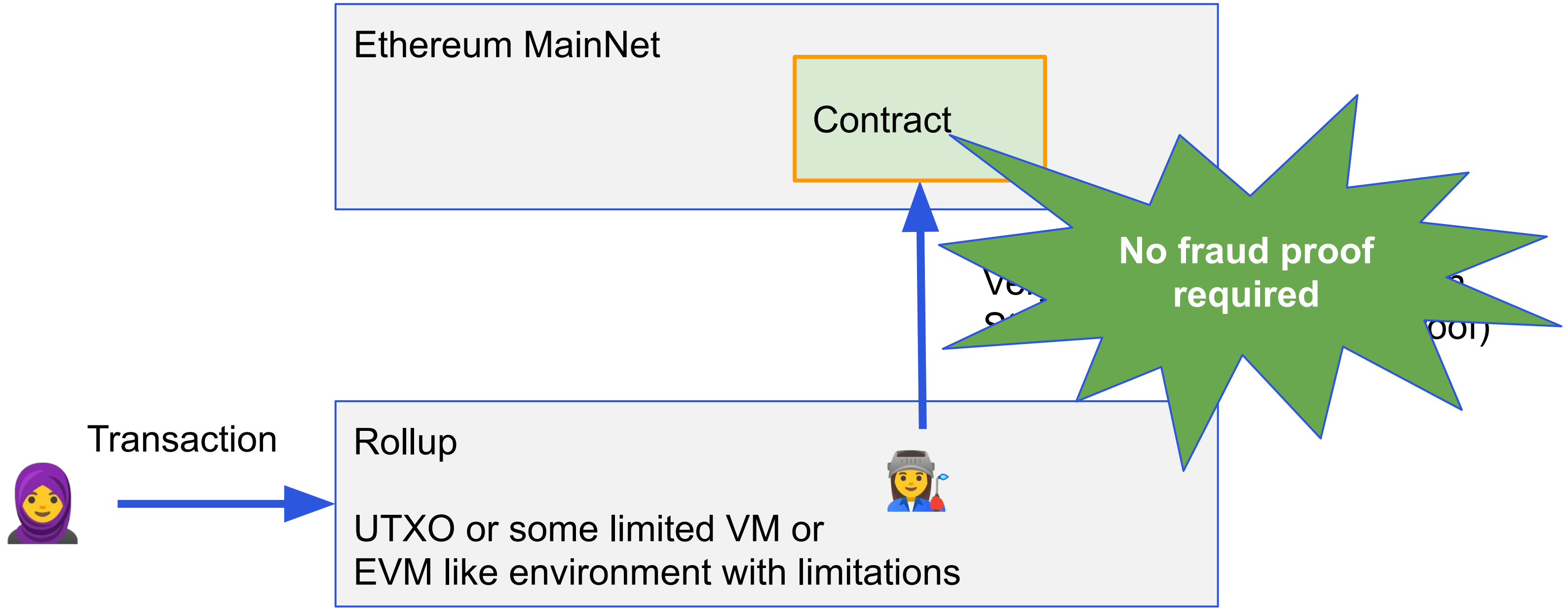


Ethereum MainNet
Contract
Compressed Transaction & State Root
Fraud Proof
Transaction
Rollup
UTXO or some limited VM or
EVM like environment with limitations

zkRollups


Ethereum MainNet
Contract
Very Compressed transaction,
State Root, zkSnark (the proof)
Transaction
Rollup
UTXO or some limited VM or
EVM like environment with limitations

zkRollups


Ethereum MainNet
Contract
No fraud proof required
Transaction
Rollup
UTXO or some limited VM or
EVM like environment with limitations

Comparison

Technology	Consensus	EVM Compatibility	Withdrawal Latency	Security Model	Disputes	Data Availabili ty
Sidechain	Separate	100%	Pin dispute window	Reputation / Trust	External: Court of Law	no
Optimistic Roll-up	Uses MainNet consensus	Mainly supported	Fraud Window	Cryptoeconmic	Fraud Proofs	yes
zkRollup	Uses MainNet consensus	More limited support	Immediate (once posted)	Cryptographic	zk	yes

Why?

Why?

People want scaling solutions to:

- Reduce transaction cost.
- Decrease latency to finality.
- Increase transaction capacity.
- Reduce state storage bloat on MainNet.
- Allow for alternative transaction models.
- Alternative consensus algorithms.
- New privacy models.
- Larger more complex transactions.
- Do research without endangering the whole of Ethereum.

Why?

People want scaling solutions to:

- **Reduce transaction cost.**
- Decrease latency to finality.
- Increase transaction capacity.
- Reduce state storage bloat on MainNet.
- Allow for alternative transaction models.
- Alternative consensus algorithms.
- New privacy models.
- Larger more complex transactions.
- Do research without endangering the whole of Ethereum.

CLICK IT OR TICKET
CARPOOLS ONLY
2 OR MORE PERSONS PER VEHICLE
Culver City NEXT 4 EXITS
EXIT 49A
ROAD WORK AHEAD
DOUBLETREE
Coach America

Sidechains

Sidechains

Peter’s definition of a Sidechain:

- A blockchain

Sidechains

Peter's definition of a Sidechain:

- A blockchain

There are a lot of forks of Geth, with the only very minor changes:

- Switch PoW (soon to be PoS) for a PoS or BFT consensus.

Are these Sidechains?

Sidechains

Peter’s definition of a Sidechain:

- A blockchain

Are consortium blockchain (GoQuorum deployments for example) Sidechains?

Sidechains

Peter's definition of a Sidechain:

- A blockchain that relies on **another blockchain** for its security.

Sidechains

Peter’s definition of a Sidechain:

- A blockchain that relies on **another blockchain** for its security.

The blockchains are separate. The could have:

- Different consensus algorithm.
- Different transaction model.
- Different execution engine.

Sidechains

Peter’s definition of a Sidechain:

- A blockchain that relies on **Ethereum MainNet** for its security.

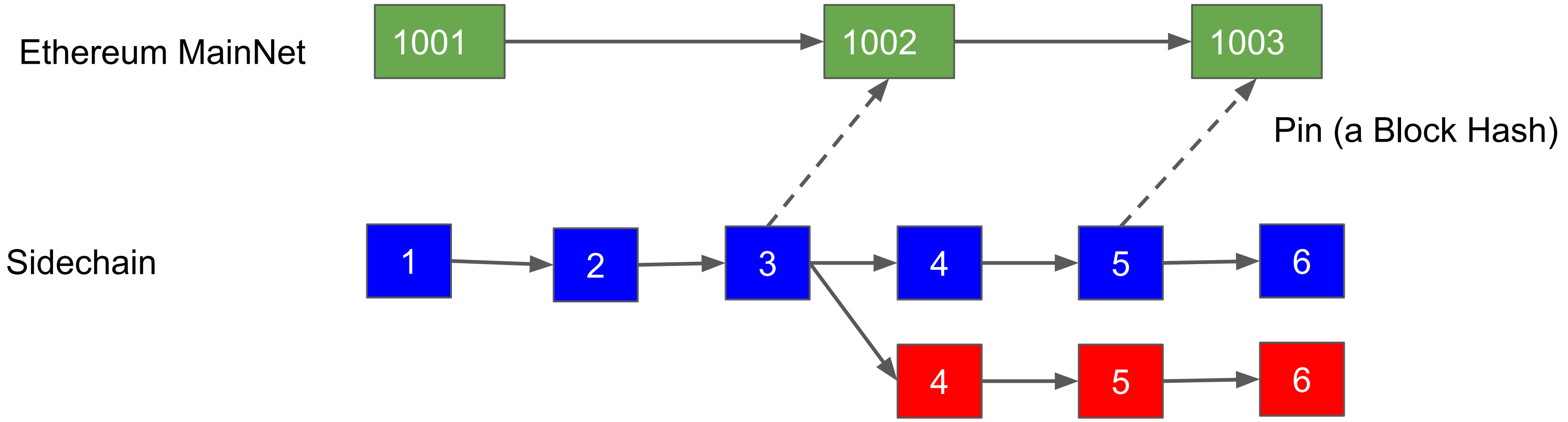
Sidechains

Peter’s definition of a Sidechain:

- A blockchain that relies on **Ethereum MainNet** for its security.

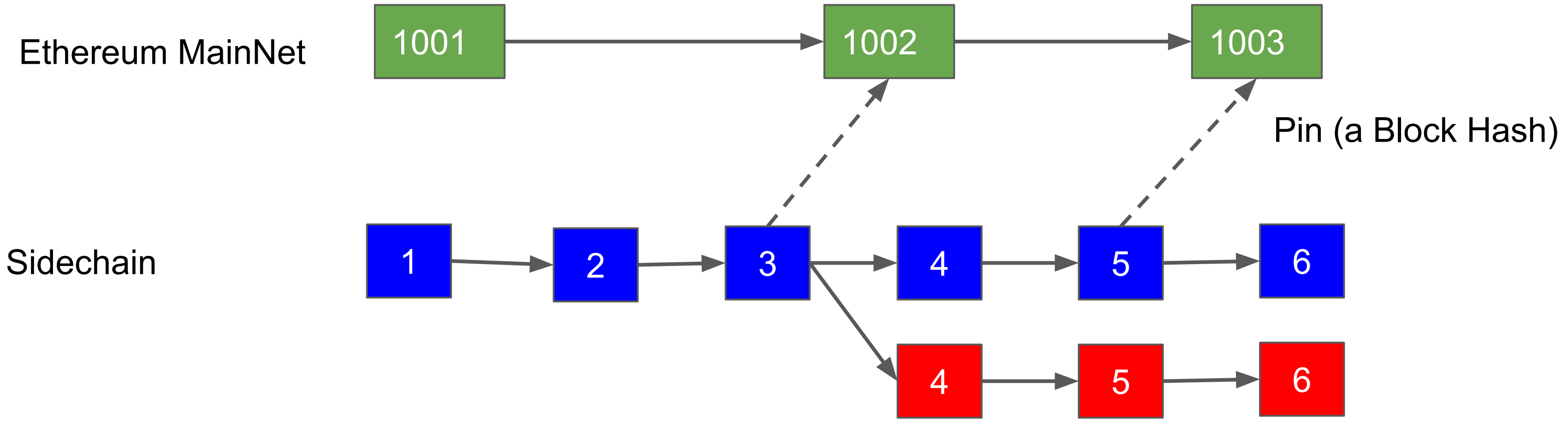
Pinning

- Security can be added to a sidechain by Pinning the state of the sidechain to another chain.



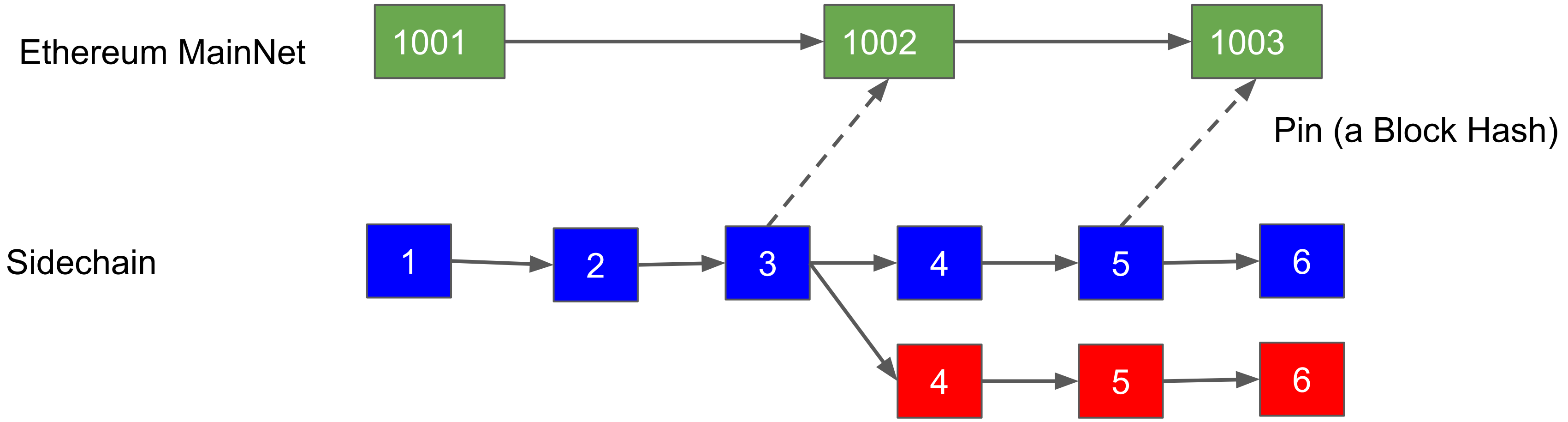
Pinning

- Pinning involves submitting the sidechain's block hash to a contract on MainNet.



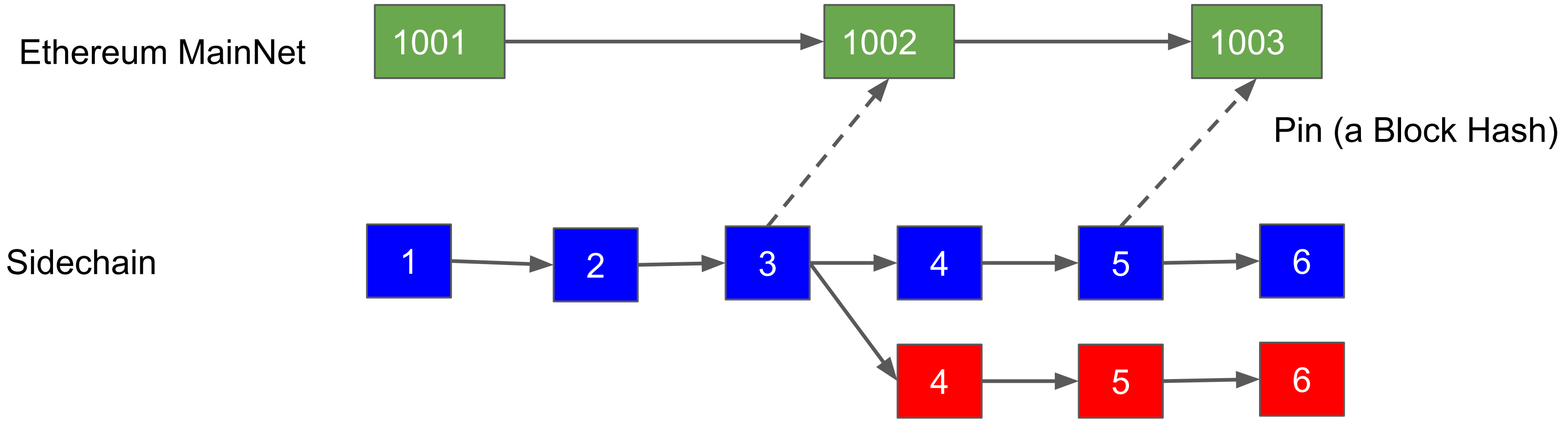
Pinning

- Pinning provides ***Collusion Resistance***: Sidechains typically do not provide adequate protection against potential collusion by consortium members to revert the state and transaction history of the blockchain.



Pinning

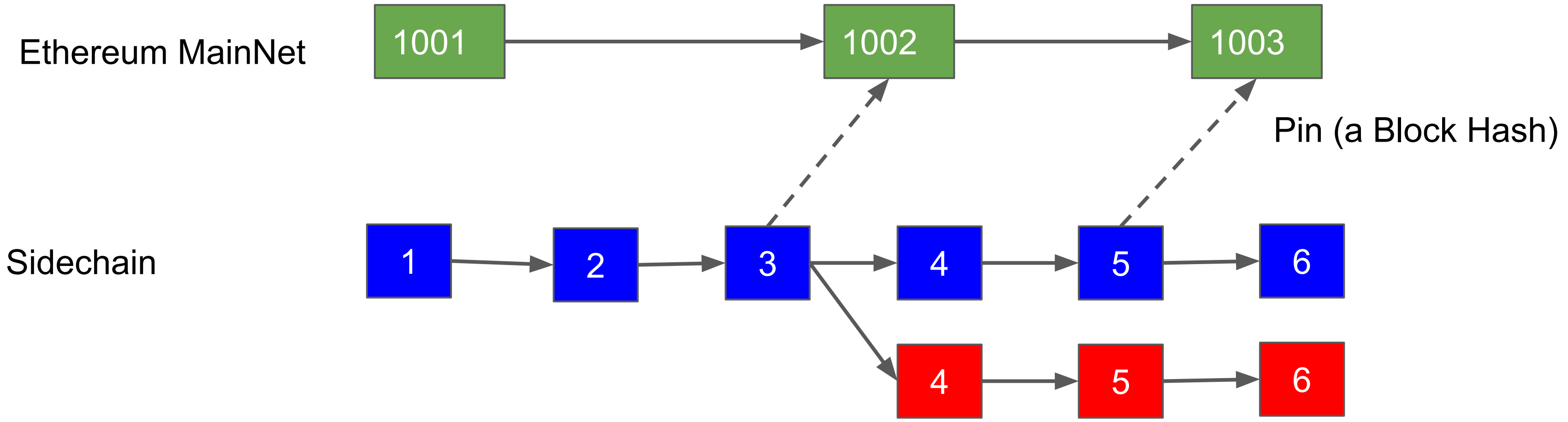
- An aggrieved party on the sidechain could show the pins and the blockchain's transaction history in a court of law to demonstrate that consortium members have colluded to revert the blockchain.



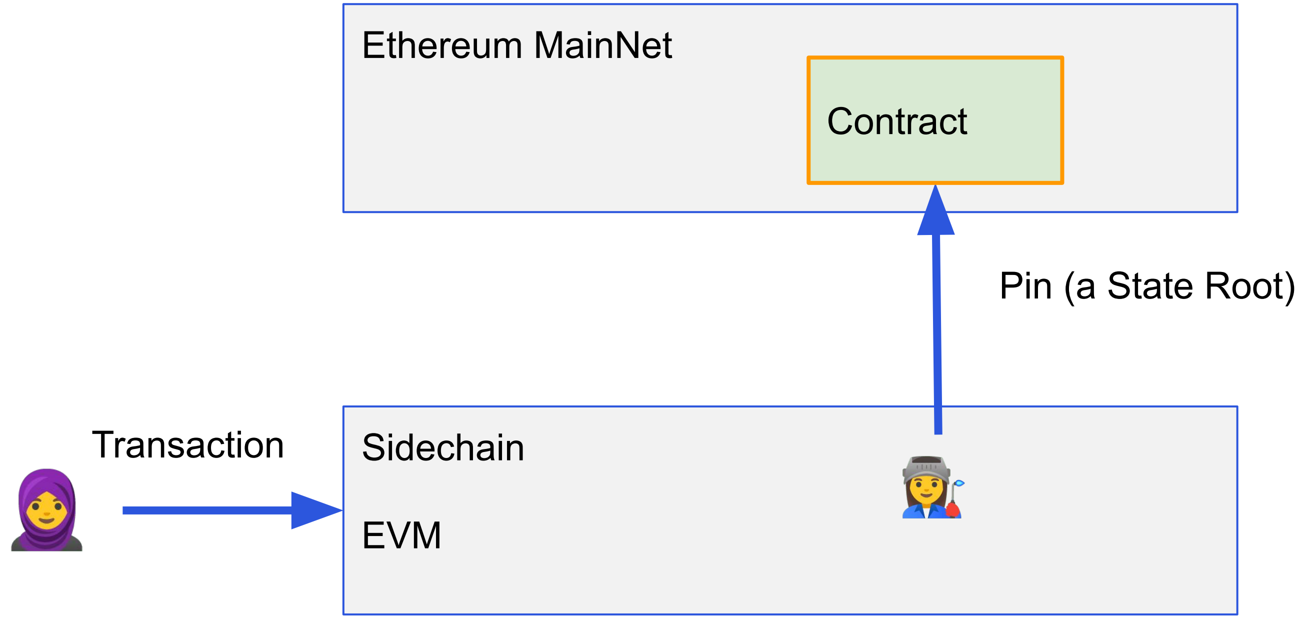
Pinning

Implementation of contract on MainNet has many options:

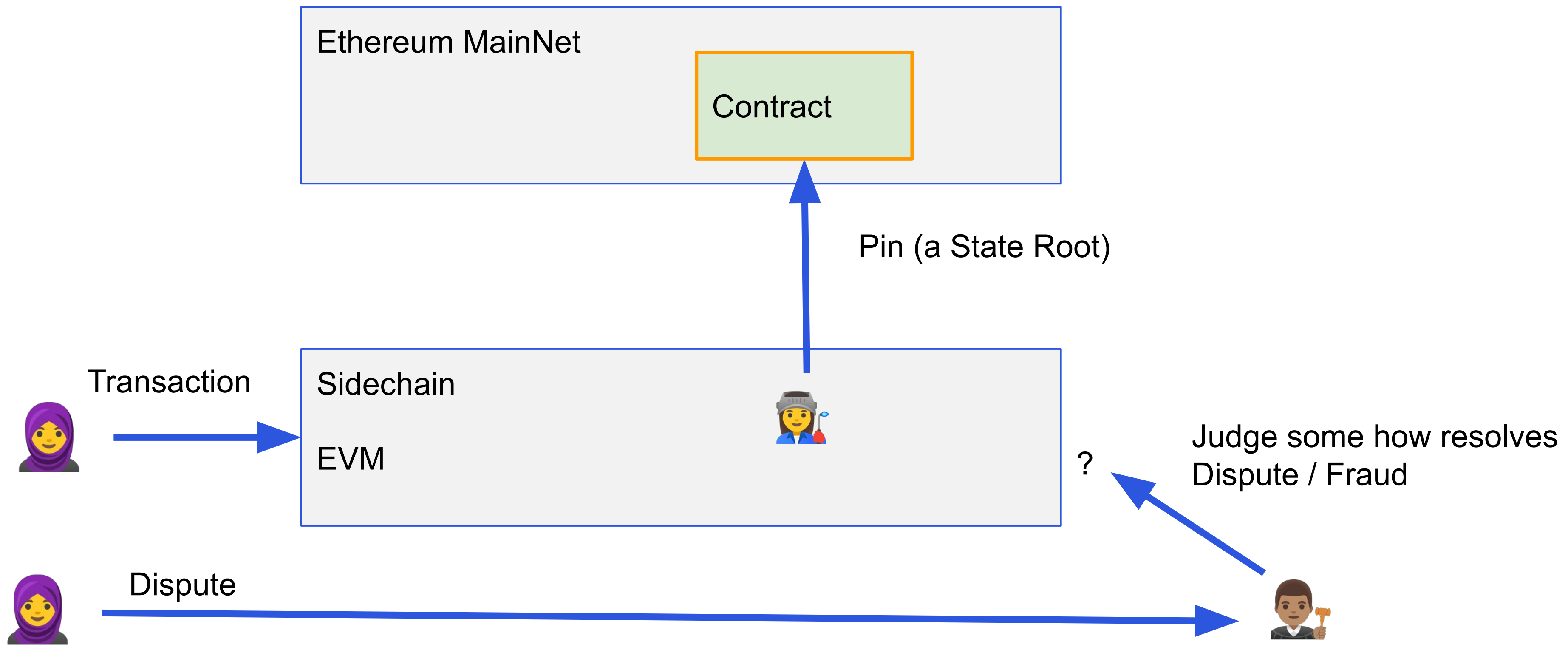
- Parties have to vote on validity of the pinned block hash.
- Multiple parties sign block hash off-chain, one party submits the multiply signed block hash, the contract on-chain only accepts the block hash if there are a threshold number of signatures.
- Use BLS threshold signing (see https://www.youtube.com/watch?v=XZTvBYG9pn4).
- Use anonymous pinning, to not reveal participants (see https://www.youtube.com/watch?v=f3EWg9aV91I).



Sidechains


Ethereum MainNet
Contract
Pin (a State Root)
Transaction
Sidechain
EVM

Sidechains


Ethereum MainNet
Contract
Pin (a State Root)
Transaction
Sidechain
EVM
?
Judge some how resolves
Dispute / Fraud
Dispute

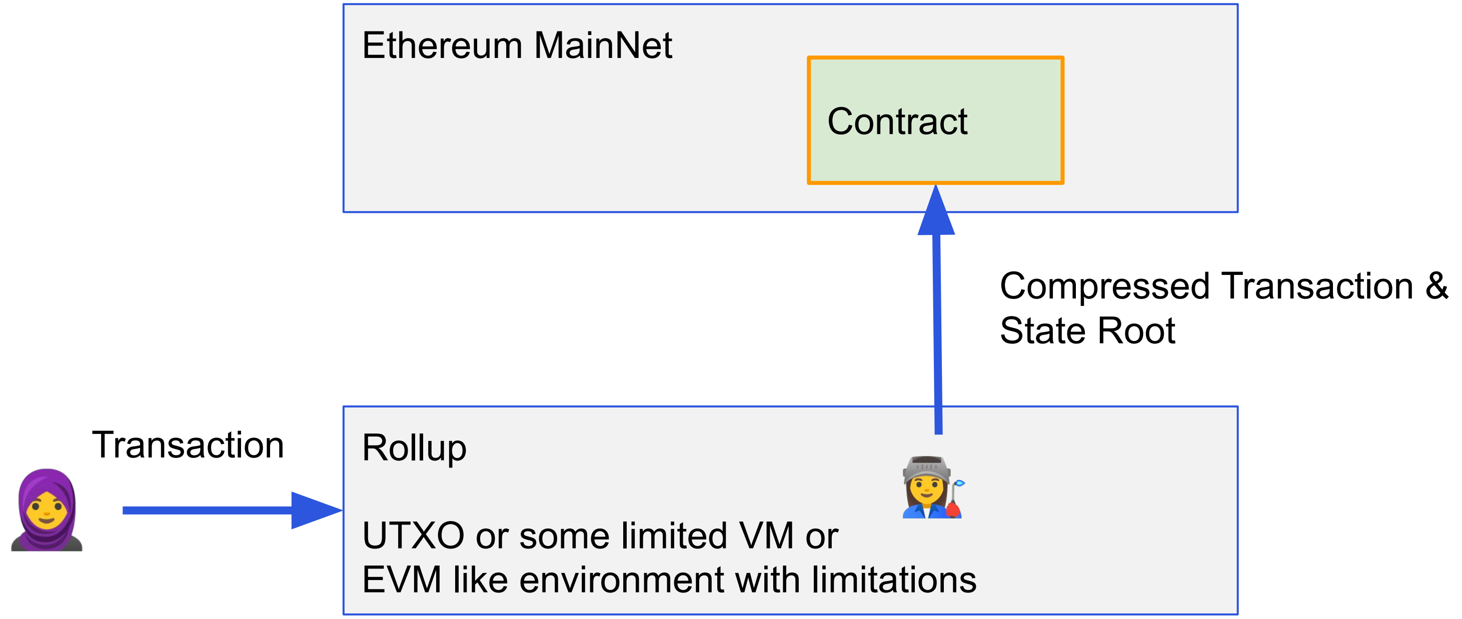
Sidechains and Pinning

Big Taken Home:

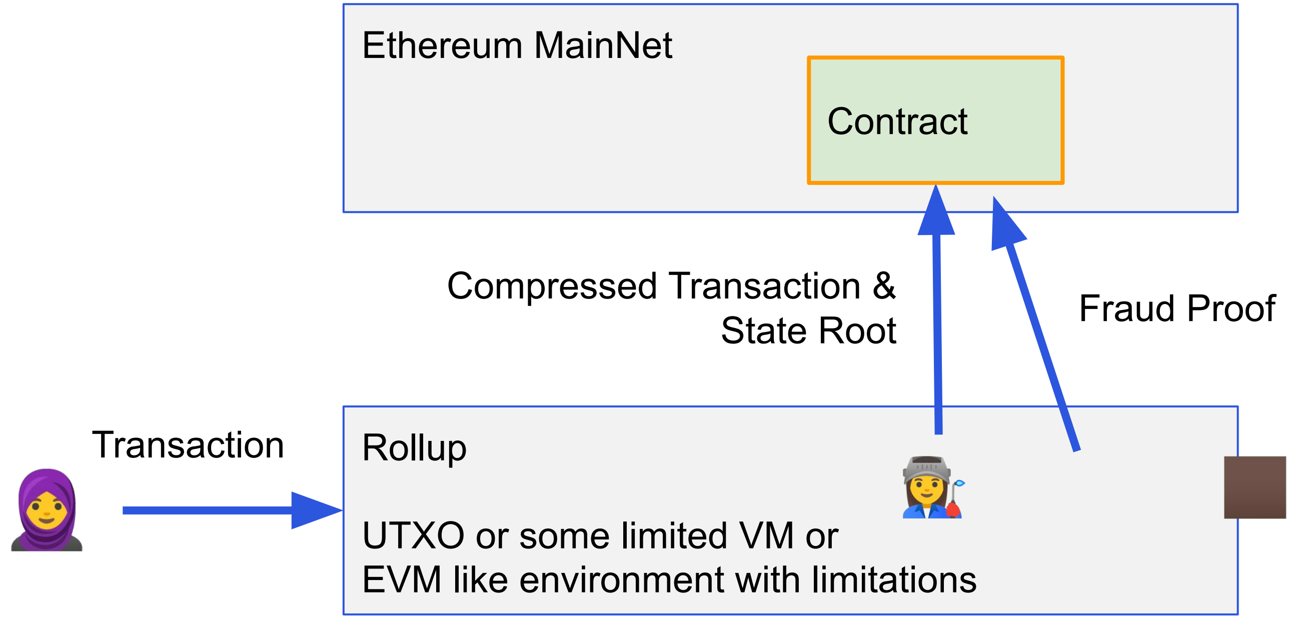
- No “data availability”. That is,
  - There is no automatic way to prove pins are valid / invalid.
- However:
  - Someone new to the sidechain could get the Genesis Block, and all transaction history, process the entire history, and verify all of the pins, thus proving to themselves that the state of the blockchain is correct (and has not been reverted).

Optimistic Rollups

Optimistic Rollups


Ethereum MainNet
Contract
Compressed Transaction & State Root
Transaction
Rollup
UTXO or some limited VM or
EVM like environment with limitations

Optimistic Rollups


Ethereum MainNet
Contract
Compressed Transaction & State Root
Fraud Proof
Transaction
Rollup
UTXO or some limited VM or EVM like environment with limitations

Optimistic Rollups

Optimistic Rollups:

- Solidity deployed on Ethereum MainNet + off-chain code that runs a set of nodes.
- **Optimistically** publish transaction results without actually executing those transactions on Ethereum MainNet (except when a fraud proof needs to execute).
- Can be Ethereum Virtual Machine (EVM) compatible code (sort of).

Optimistic Rollups

Fuel

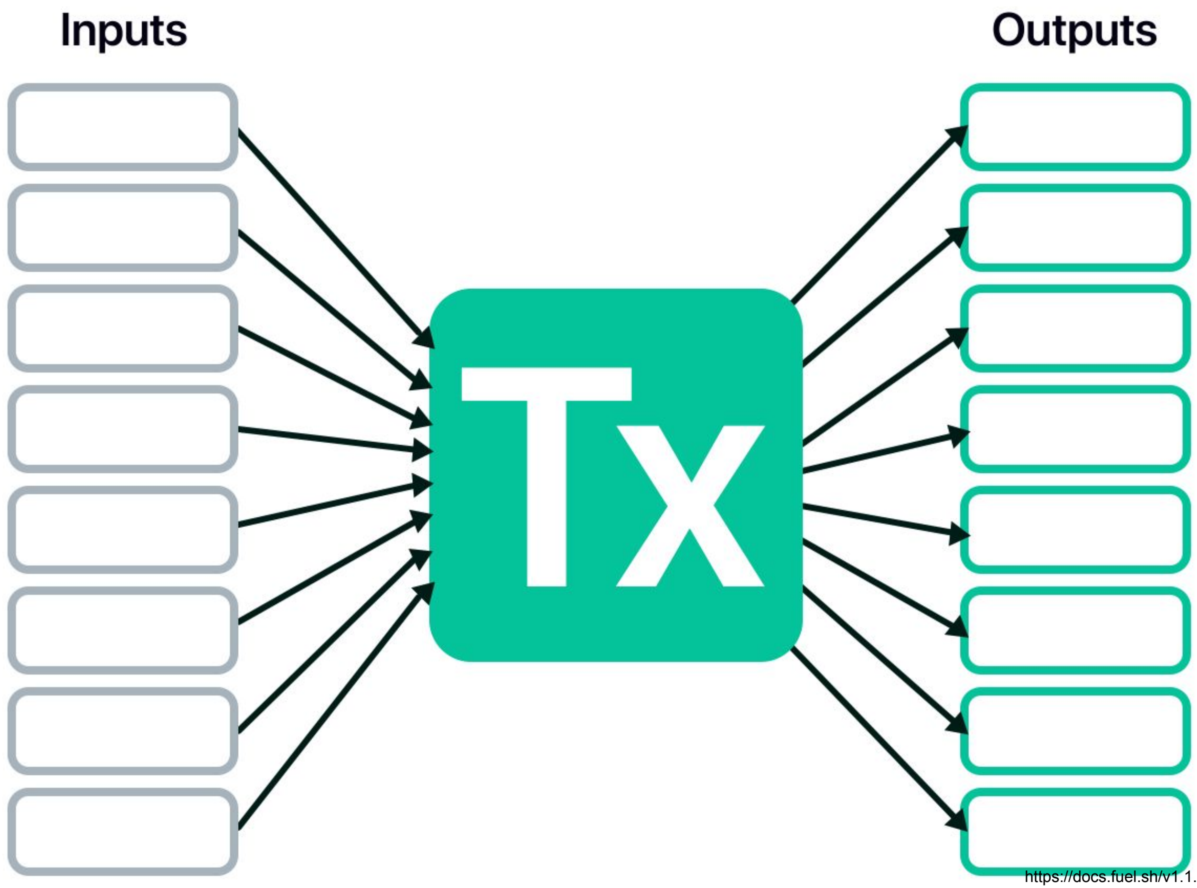
Optimism

ARBITRUM

Fuel

https://docs.fuel.sh/v1.1.0/Concepts/Fundamentals/Fuel%20Overview.html

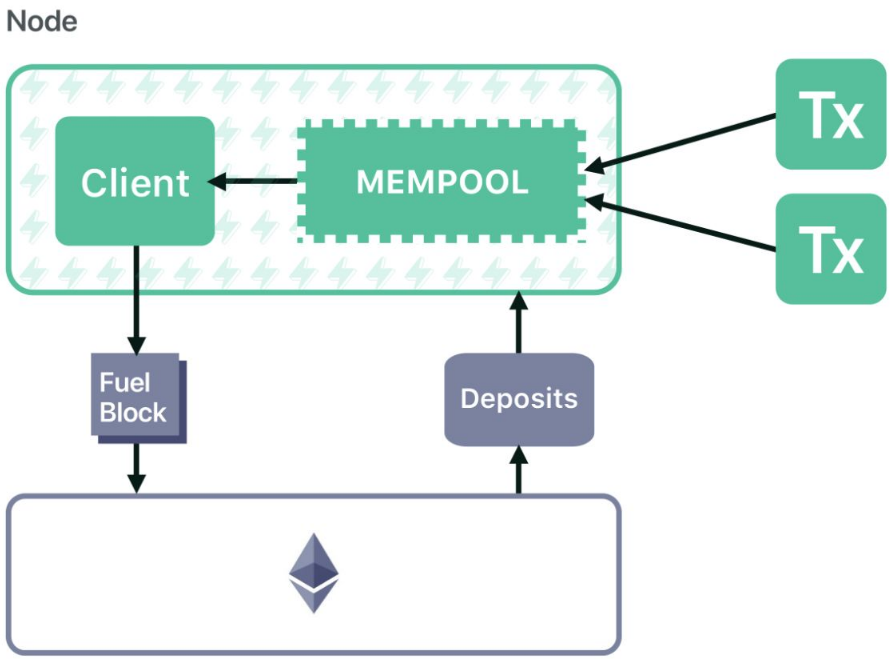
Fuel: UTXO



https://docs.fuel.sh/v1.1.0/Concepts/Fundamentals/Fuel%20Overview.html

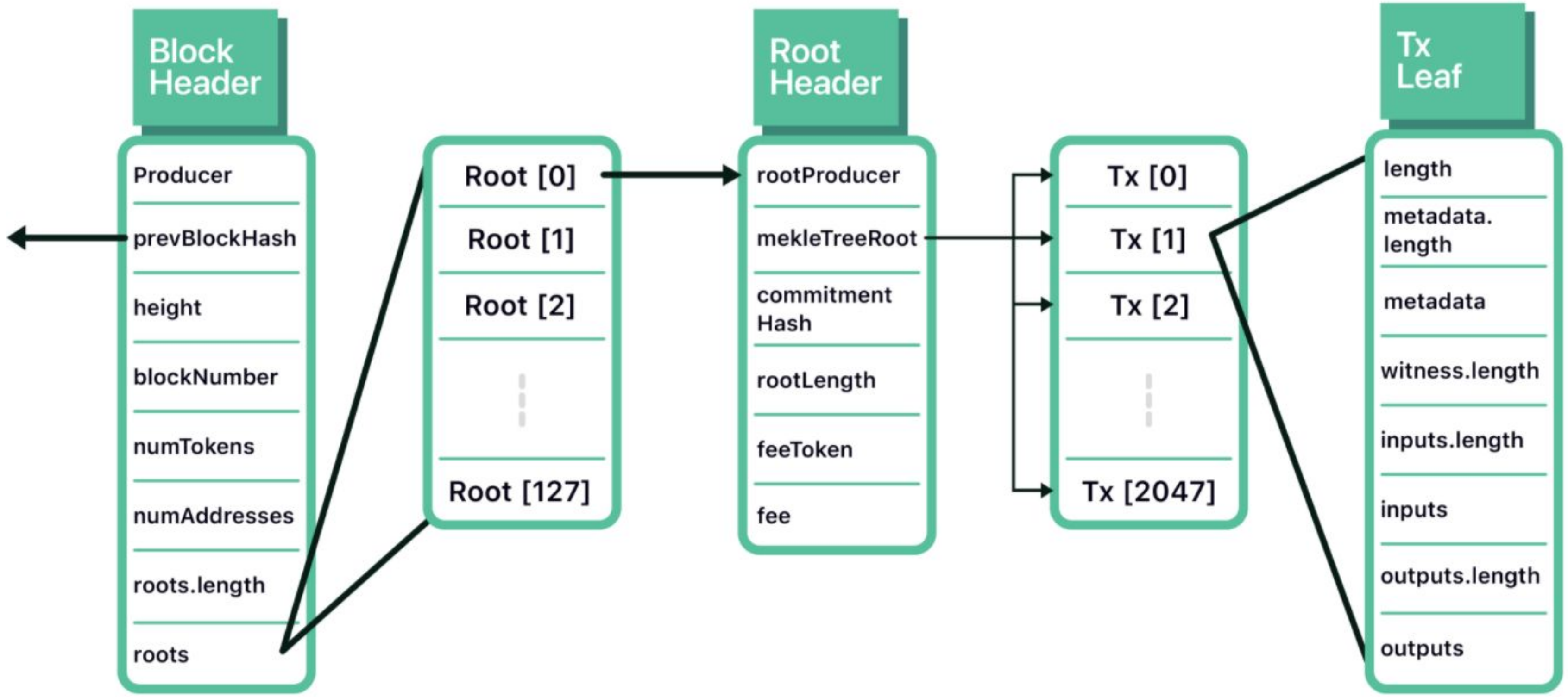
Fuel

- Anyone may construct a rollup block off-chain and submit it to Fuel contract on Ethereum MainNet.
- Each block must build upon the previous rollup block (block header hashes) & include a bond.
- If a block is invalid, a compact fraud proof can be submitted and processed on-chain by anyone.
  - Rolls back the rollup chain to the previous block.
  - Burns a portion of the bond.
  - Rewards the fraud prover with the rest.
- After a timeout, rollup blocks are finalized, are considered valid and their bond unlocked.
- Withdrawals are initiated by burning coins on the rollup, then completing the withdrawal after that rollup block has finalized.



https://docs.fuel.sh/v1.1.0/Concepts/Fundamentals/Fuel%20Overview.html

Fuel



Source: https://docs.fuel.sh/v1.1.0/Concepts/Fundamentals/Block%20Architecture.html

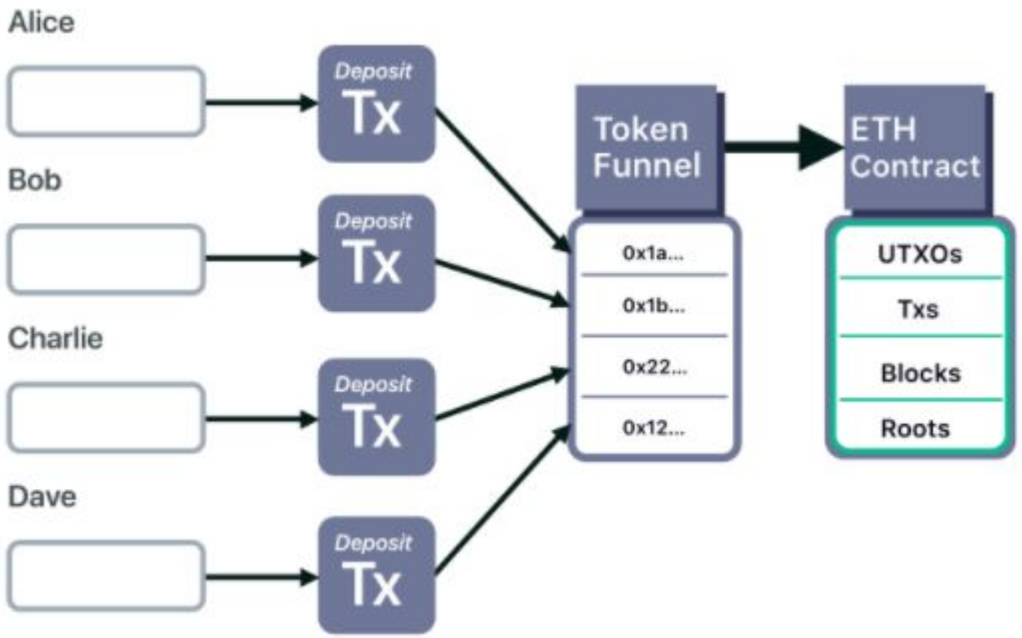
Fuel: Transfer Tokens -> Fuel Rollup

Can ERC 20 approve, then transfer.

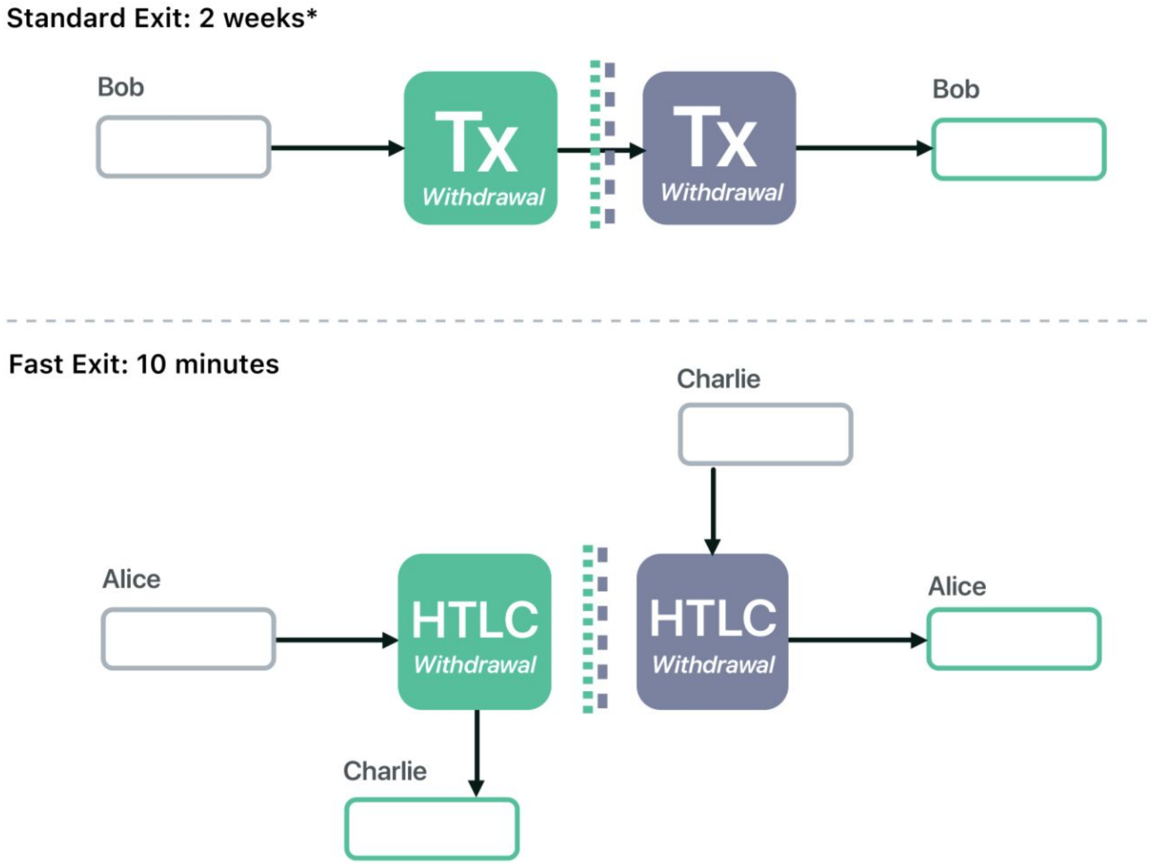
OR

Token funnel: Transfer to address and then use CREATE2 to create a contract at the address, do the transfer, and then self-destruct.

A node of the Fuel Rollup notices the transfer and then mints the token on the Roll-up.



Fuel: Exit



Standard Exit: 2 weeks*
Bob
Tx
Withdrawal
Tx
Withdrawal
Bob
Fast Exit: 10 minutes
Charlie
Alice
HTLC
Withdrawal
HTLC
Withdrawal
Alice
Charlie

Fuel

Notes:

- Supports coloured coins.
- Explorer: https://mainnet.fuel.sh/network
- V2 will bring a VM (not EVM compatible).

Optimism

Optimism (https://optimism.io/)

All in one page (sort of):

- Users submit transactions to bonded **Sequencers**.
- For each transaction, the Sequencers:
    - Create an OVM Message, a compressed for of the transaction.
    - Execute the OVM message using the Optimism Virtual Machine on the Rollup (on L2).
    - As part of the execution, state may be stored on the Rollup.
    - A new State Root is calculated after each transaction / OVM Message.
- The Sequencer publishes a batch of transactions and State Roots to Optimism's contracts on Ethereum MainNet: OVM_CanonicalTransactionChain contract, and the OVM_StateCommitmentChain contract.
- Anyone can submit a fraud proof to the OVM_FraudVerifier contract on Ethereum MainNet to revert a bad state update (and all subsequent state updates) (note: not reverting transactions).
- Slashing:
    - Sequencers that publish fraudulent State Roots.
    - Sequencers that build off a fraudulent State Roots.
- Fraud Proof Window is 1 week.

Optimism's OVM Messages

Ethereum Transactions		OVM Messages	
To	20 bytes	To	20 bytes
Nonce	32 bytes	Nonce	3 bytes
Gas Limit	32 bytes	Gas Limit	3 bytes
Gas Price	32 bytes	Gas Price	3 bytes
Signature R	32 bytes	Signature R	32 bytes
Signature S	32 bytes	Signature S	32 bytes
Signature V	1 byte	Signature V	1 byte
Payload	? bytes	Payload	? bytes
Amount	32 bytes		

Optimism’s OVM Messages

OVM Messages	
To	20 bytes
Nonce	3 bytes
Gas Limit	3 bytes
Gas Price	3 bytes
Signature R	32 bytes
Signature S	32 bytes
Signature V	1 byte
Payload	? bytes
Signature type	1 byte

OVM Message fields are put into the Payload of a transaction.

The transaction is submitted to the sequencer.

The OVM message and other metadata

The translation from the original transaction to the OVM message inside the new transaction happens inside of Optimism’s fork of Geth.

Optimism Virtual Machine: OVM

- OVM execution is very similar to EVM execution.
- Some EVM instructions are not available.
- Some EVM instructions are available by calling into a the OVM_ExecutionManager contract.
- Forked Solidity compiler used to create the appropriate code.

Optimism Virtual Machine: Instructions not available

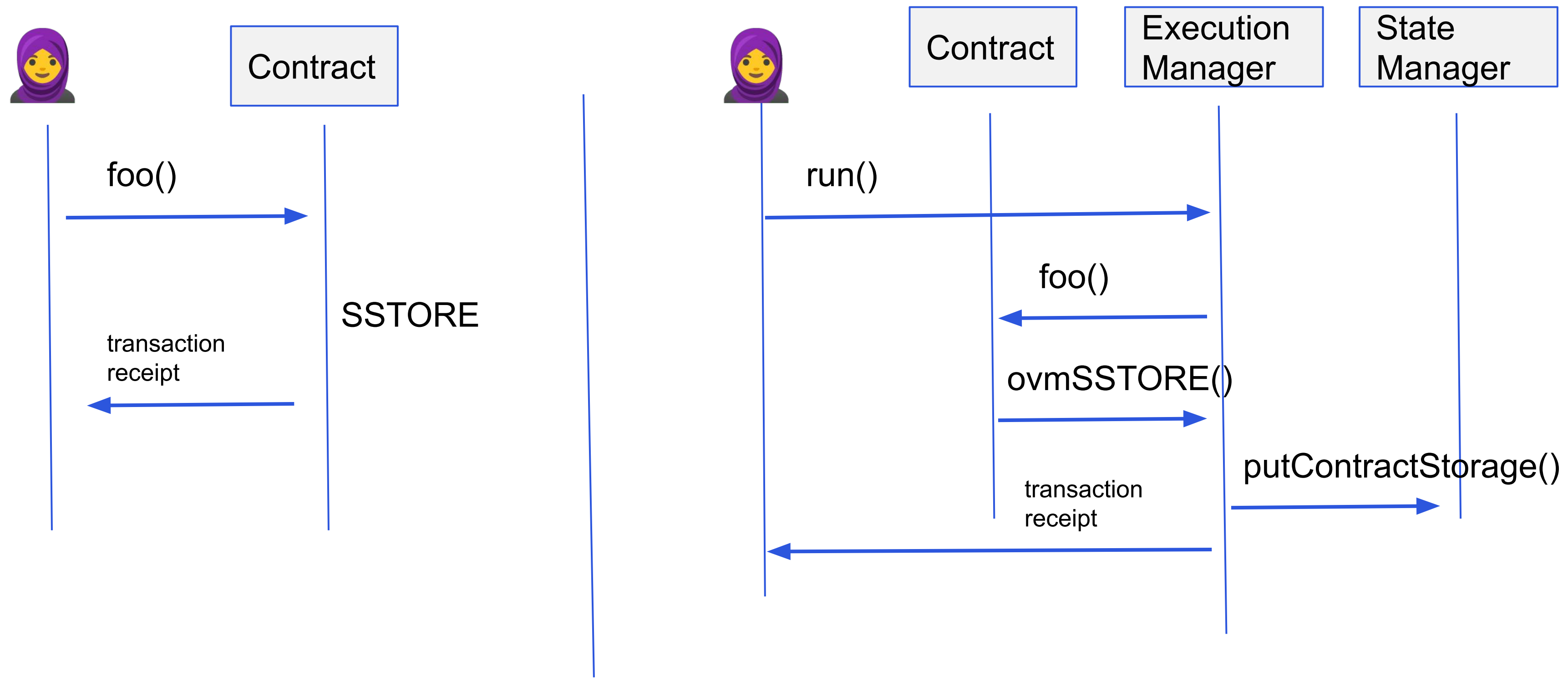
EVM Opcode	Solidity Usage
`COINBASE`	`block.coinbase`
`DIFFICULTY`	`block.difficulty`
`BLOCKHASH`	`blockhash`
`GASPRICE`	`tx.gasprice`
`SELFDESTRUCT`	`selfdestruct`
`BALANCE`	`balance`
`CALLVALUE`	`msg.value`
`ORIGIN`	`tx.origin`

Optimism Virtual Machine: Implemented via ExecutionManager

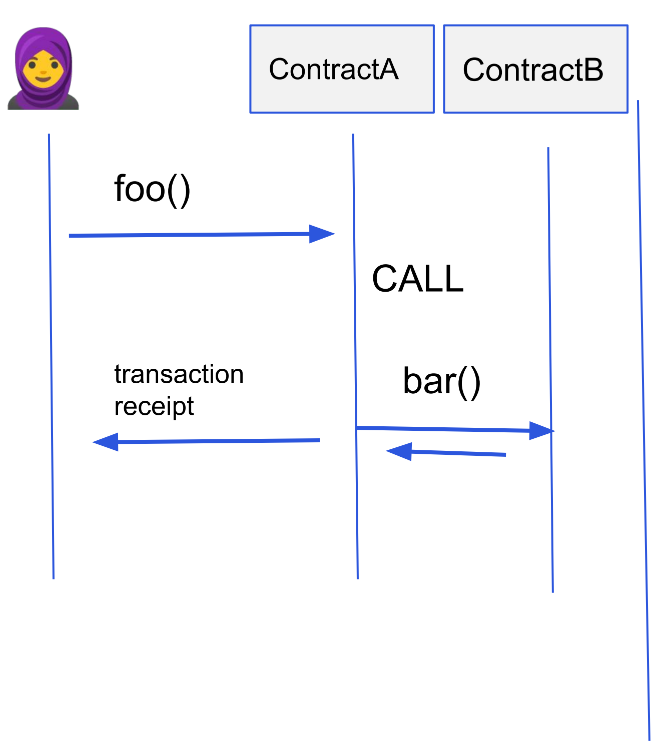
EVM Opcode	OVM Equivalent
ADDRESS	ovmADDRESS
NUMBER	ovmNUMBER
TIMESTAMP	ovmTIMESTAMP
CHAINID	ovmCHAINID
GASLIMIT	ovmGASLIMIT
REVERT	ovmREVERT
SLOAD	ovmSLOAD
SSTORE	ovmSSTORE

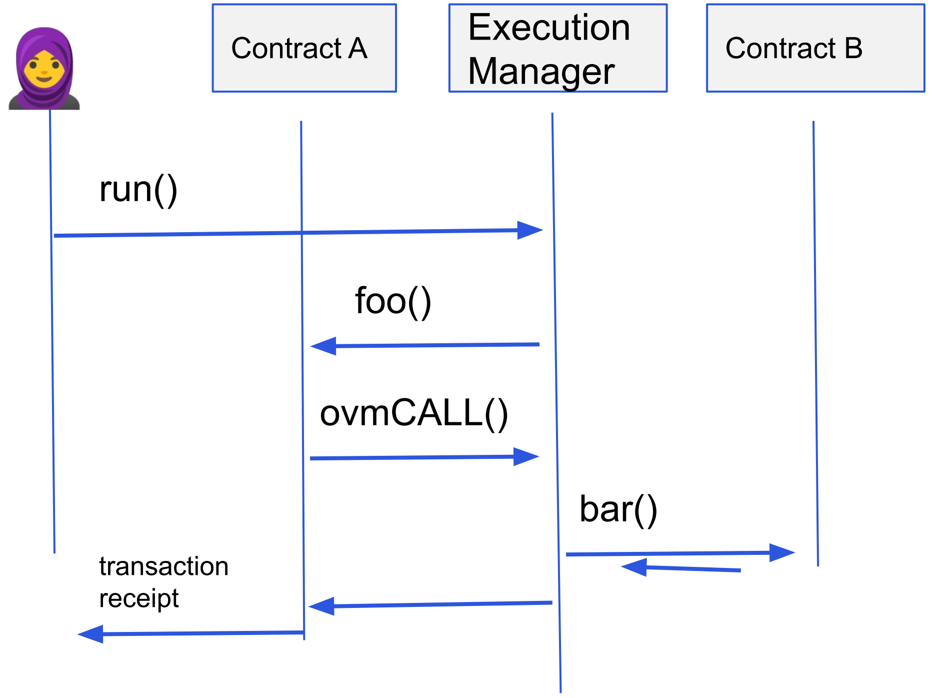
EVM Opcode	OVM Equivalent
CALL	ovmCALL
STATICCALL	ovmSTATICCALL
DELEGATECALL	ovmDELEGATECALL
CREATE	ovmCREATE
CREATE2	ovmCREATE2
EXTCODECOPY	ovmEXTCODECOPY
EXTCODESIZE	ovmEXTCODESIZE
EXTCODEHASH	ovmEXTCODEHASH

Optimism: Instruction Translation



Optimism: Instruction Translation





OVM Safety Checker

OVM Safety Checker:

- Ensures code doesn’t include any unsupported opcodes.

However, if you try hard you can bypass it and crash your transaction:

- If code is passed in as a parameter to a constructor, and that code contains unsupported opcodes, you will receive an error.

Optimism Storage: OVM_StateManager

State Manager:

- Holds state as bytes32 of all contracts on Roll-up:
  - state[contract address][storage location] = value

On the Roll-up:

- State Manager implemented as a pre-compile.
- Holds all state.

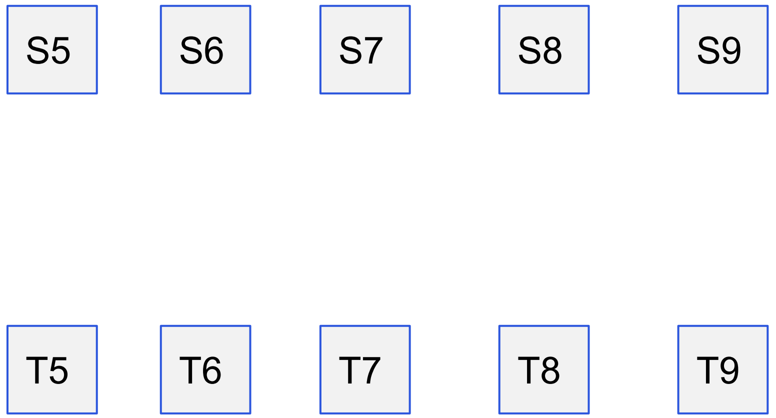
Om Ethereum, MainNet:

- Solidity Contract deployed on MainNet.
- Used with fraud proof checking.

Optimism: Fraud Proofs

OVM_StateCommitmentChain contract

OVM_CanonicalTransactionChain contract

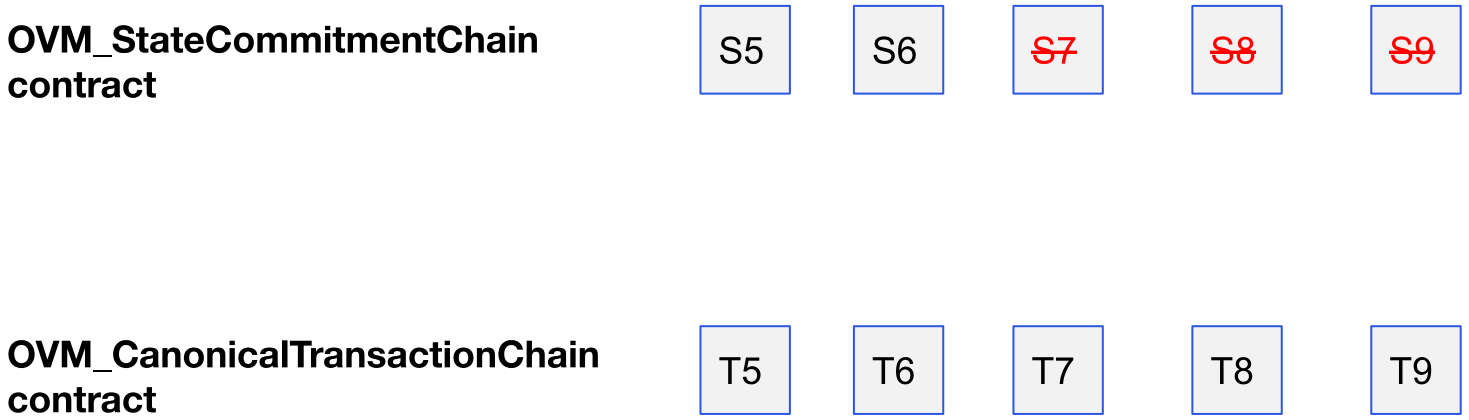


Optimism: Fraud Proofs

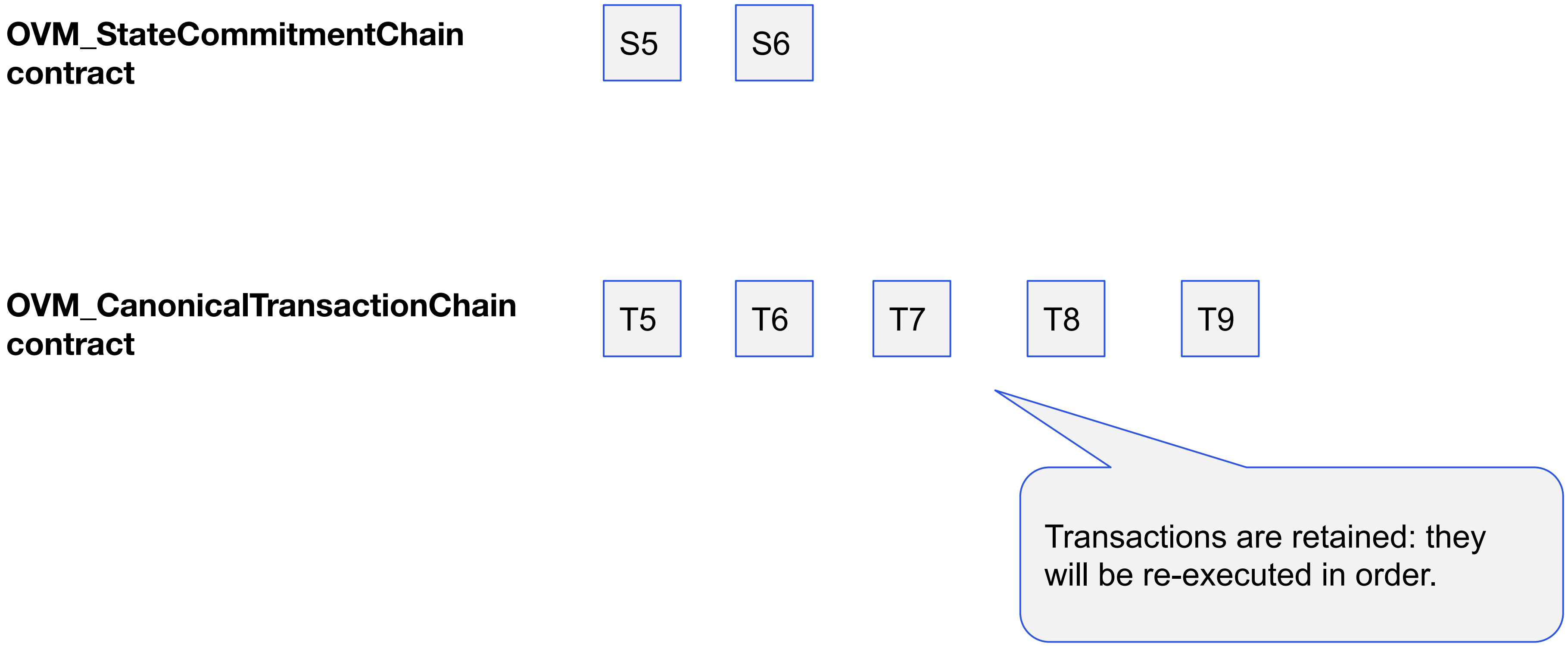
Submit a transactions on Ethereum MainNet to OVM_FraudVerifier.sol:

- initializeFraudVerification: State which transaction / state change is being disputed.
- Create the state:
  - Deploy contracts on MainNet.
  - Link deployed contract with contract address from L2.
  - Set relevant state (supplied along with Merkle Proofs to show it is part of the state).
- Run the transaction. This will update the state.
- Supply Merkle Proofs for the parts of state that are in the Rollup, but aren't available in the partial state tree on MainNet.
- finalizeFraudVerification: Prove that the calculated state root does not match the posted state root.

Optimism: Fraud Proofs


OVM_StateCommitmentChain contract
S5
S6
~~S7~~
~~S8~~
~~S9~~
OVM_CanonicalTransactionChain contract
T5
T6
T7
T8
T9

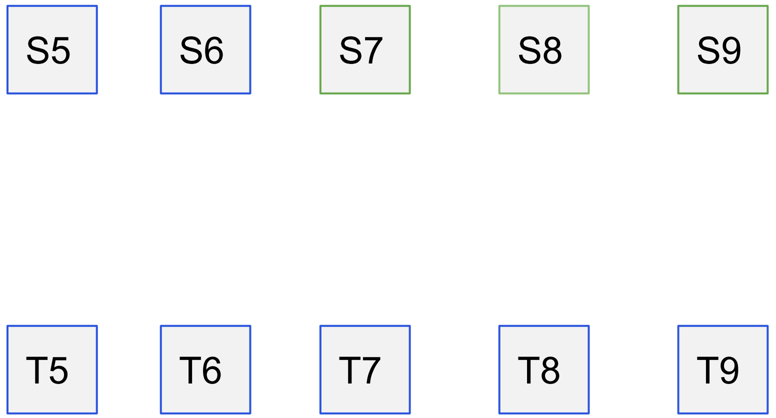
Optimism: Fraud Proofs


OVM_StateCommitmentChain contract
S5
S6
OVM_CanonicalTransactionChain contract
T5
T6
T7
T8
T9
Transactions are retained: they will be re-executed in order.

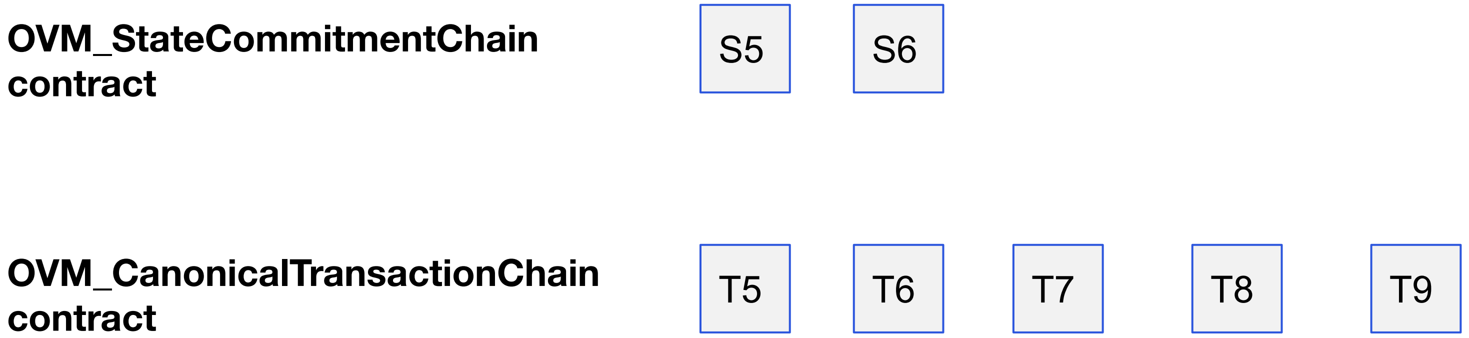
Optimism: Fraud Proofs

OVM_StateCommitmentChain contract

OVM_CanonicalTransactionChain contract



Optimism Fraud Proof Verification in Parallel

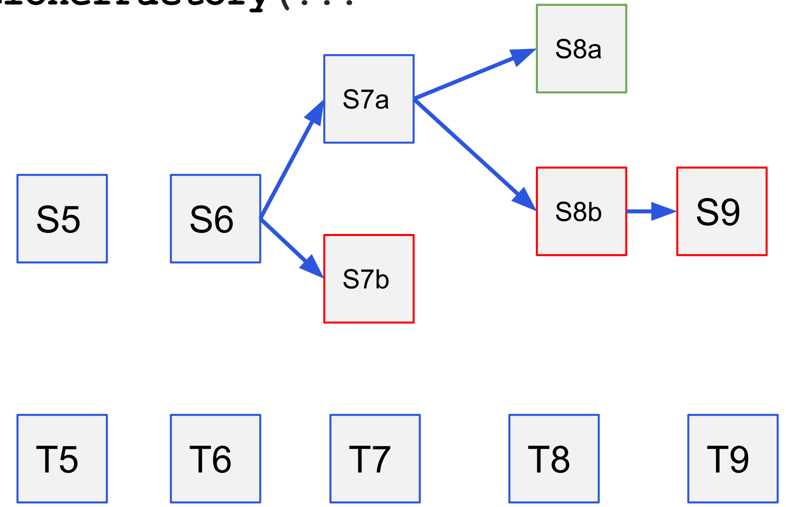

OVM_StateCommitmentChain contract
S5
S6
OVM_CanonicalTransactionChain contract
T5
T6
T7
T8
T9

Optimism Fraud Proof Verification in Parallel

```
transitioners[keccak256(abi.encodePacked(_preStateRoot, _txHash))] =
          iOVM_StateTransitionerFactory(...
```

OVM_StateCommitmentChain contract

OVM_CanonicalTransactionChain contract



See: https://github.com/ethereum-optimism/optimism/blob/develop/packages/contracts/contracts/optimistic-ethereum/OVM/verification/OVM_FraudVerifier.sol

Ether on Optimism

- Represented as an ERC 20 contract on Roll-up at address
  `0x4200000000000000000000000000000000000006`
- ethGetBalance returns balance for account based on what is in ERC 20 contract.

Gas on Optimism

- Transactions that you send directly to the rollup smart contracts are paid for by burning a small amount of gas on Ethereum.
- Sequencers and transaction aggregators may additionally charge fees for their services. Fees for these services are typically charged by sending the service providers a small amount of ETH on Layer 2.

Transfer to / from Optimism

- OVM Messages can be sent between Layer 1 & 2.
- These are function calls across blockchains.
- However, they do not provide atomic updates (don’t provide the *safety* property).
    - That is, the function call on the receiving blockchain / rollup could fail.

Transfer to / from Optimism

L1 to L2:

- For example, to move an ERC 20 token from L1 to L2:
    - On L1, transfer ERC 20 to the Roll-up contract
    - Transaction occurs, emits an event with crosschain OVM message, plus puts message in an outgoing transaction queue.
    - OVM Message forwarded to L2.
    - On L2, mint some new ERC 20 for the user.
- Appears to relying on relayers to forward messages.
- Appears to not be atomic / might have safety issues.
    - That is, if the OVM message executed on L2 fails for some reason, the L1 function isn't going to know about it.

L2 to L1:

- Via the published OVM messages.
- Requires waiting one week for the Fraud Proof Window to expire.
- Fraud Proof window needed to ensure censorship resistance on Ethereum MainNet.

Optimism with HTLC or GPACT

To get around the Fraud Proof Window and to provide atomic behaviour, what about?

- Hash Timelocked Contracts (HTLC)s
- General Purpose Atomic Crosschain Transactions (GPACT)

Optimism with HTLC

HTLC on other Blockchain

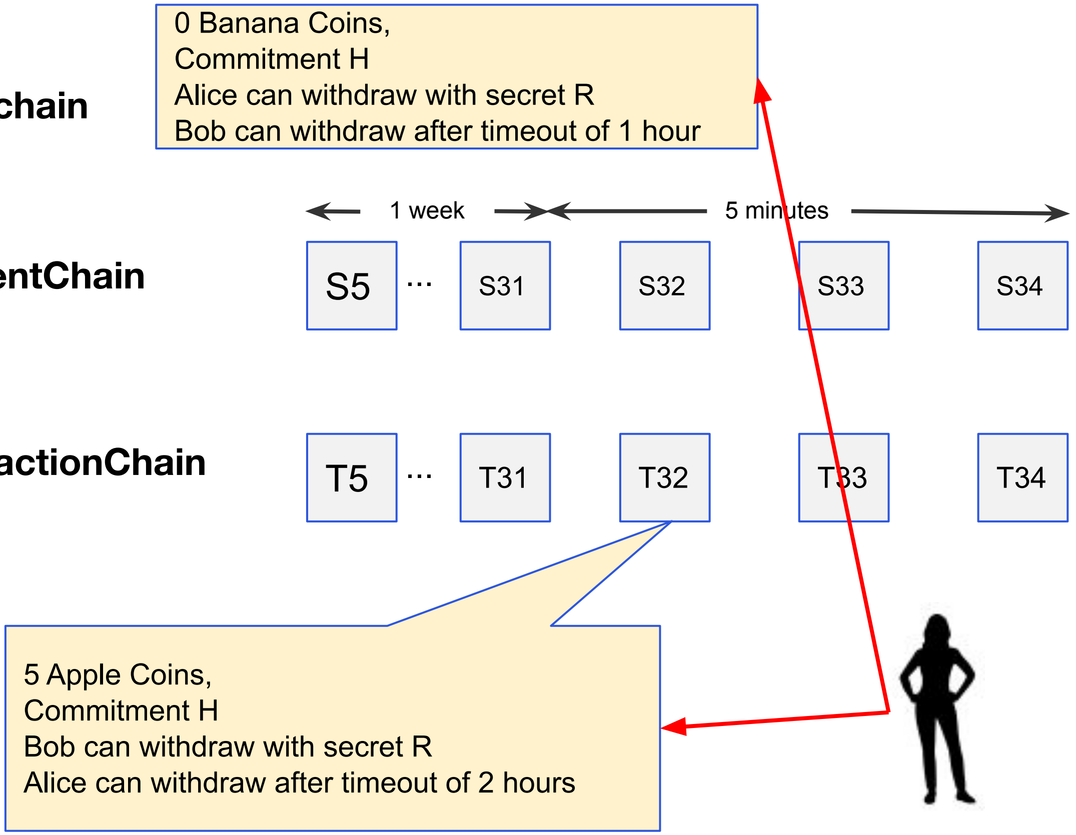
0 Banana Coins,
Commitment H
Alice can withdraw with secret R
Bob can withdraw after timeout of 1 hour

OVM_StateCommitmentChain contract

OVM_CanonicalTransactionChain contract

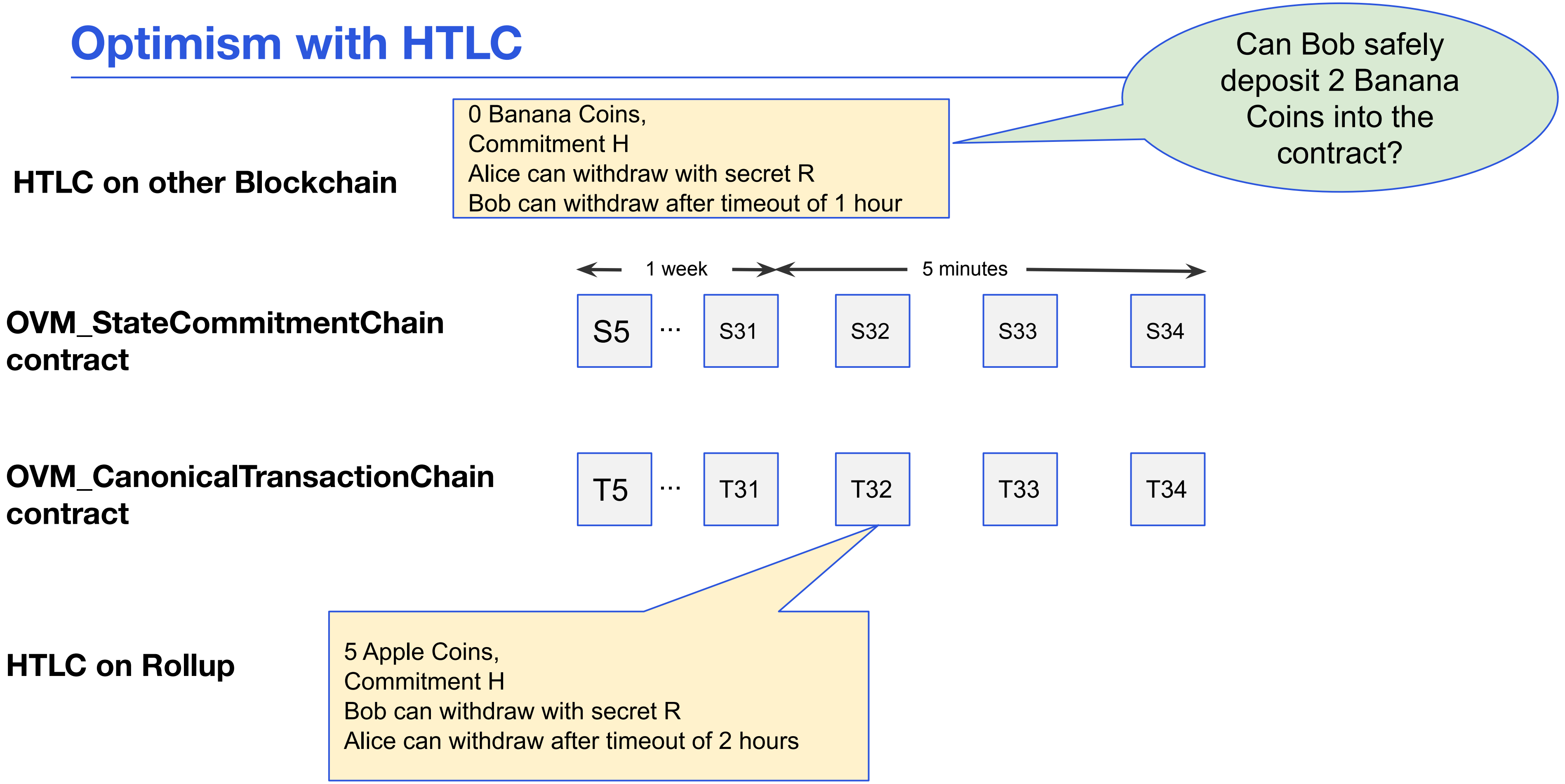
HTLC on Rollup

5 Apple Coins,
Commitment H
Bob can withdraw with secret R
Alice can withdraw after timeout of 2 hours



Alice deploys a HTLC on the Rollup and on another blockchain

Optimism with HTLC



0 Banana Coins,
Commitment H
Alice can withdraw with secret R
Bob can withdraw after timeout of 1 hour

Can Bob safely deposit 2 Banana Coins into the contract?

5 Apple Coins,
Commitment H
Bob can withdraw with secret R
Alice can withdraw after timeout of 2 hours

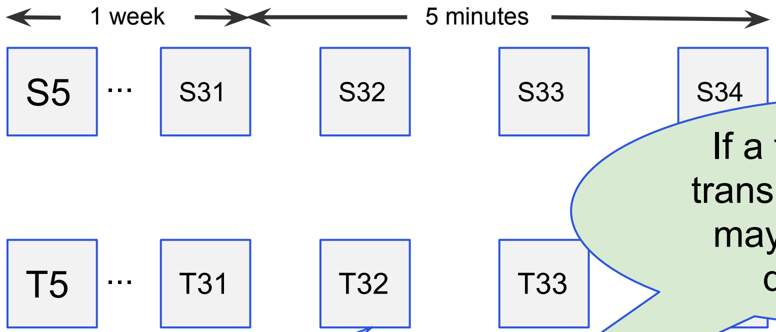
Optimism with HTLC

HTLC on other Blockchain

0 Banana Coins,
Commitment H
Alice can withdraw with secret R
Bob can withdraw after timeout of 1 hour

OVM_StateCommitmentChain contract

OVM_CanonicalTransactionChain contract



If a fraudulent state transition was posted, maybe the contract doesn't exist.

HTLC on Rollup

5 Apple Coins,
Commitment H
Bob can withdraw with secret R
Alice can withdraw after timeout of 2 hours

Even if the deployment transaction exists, maybe Alice didn't have enough $ to deploy the contract.

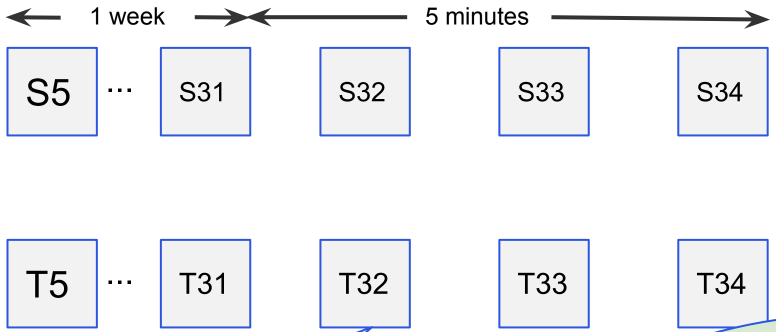
Optimism with HTLC

HTLC on other Blockchain

0 Banana Coins,
Commitment H
Alice can withdraw with secret R
Bob can withdraw after timeout of 1 hour

OVM_StateCommitmentChain contract

OVM_CanonicalTransactionChain contract



HTLC on Rollup

5 Apple Coins,
Commitment H
Bob can withdraw with secret R
Alice can withdraw after timeout of 2 hours

Bob could combine S5 with T6 to T34 to confirm that S34 is valid, and that the contract has been deployed.

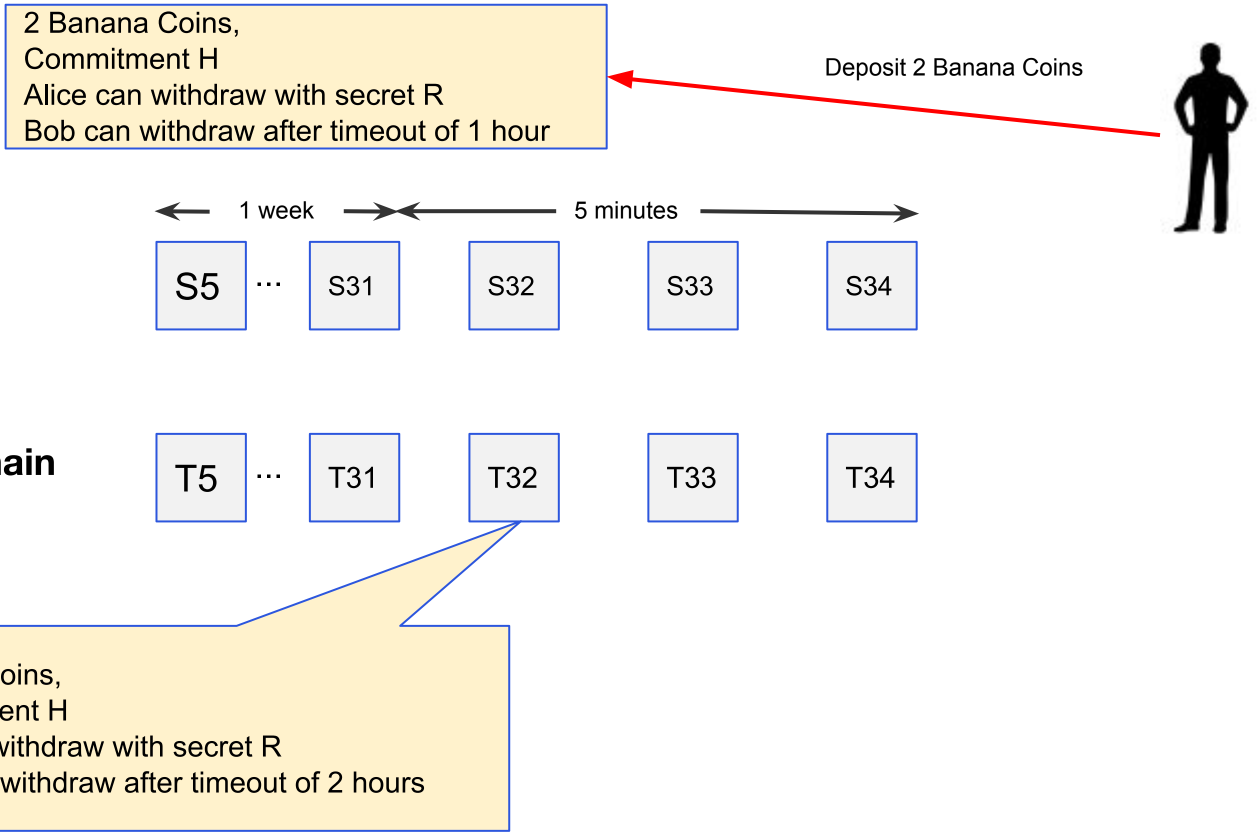
Optimism with HTLC

HTLC on other Blockchain

OVM_StateCommitmentChain contract

OVM_CanonicalTransactionChain contract

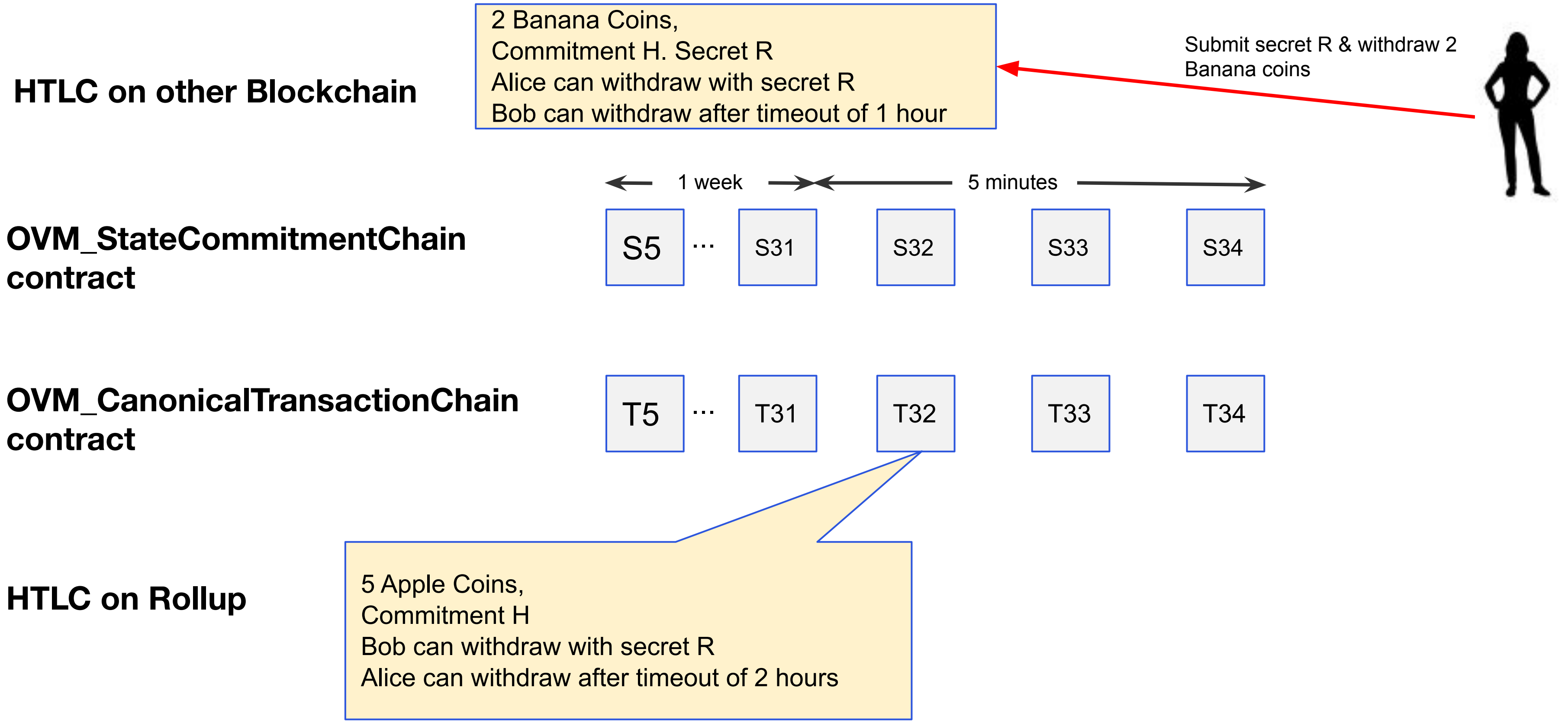
HTLC on Rollup



2 Banana Coins,
Commitment H
Alice can withdraw with secret R
Bob can withdraw after timeout of 1 hour

5 Apple Coins,
Commitment H
Bob can withdraw with secret R
Alice can withdraw after timeout of 2 hours

Optimism with HTLC



HTLC on other Blockchain

2 Banana Coins,
Commitment H. Secret R
Alice can withdraw with secret R
Bob can withdraw after timeout of 1 hour

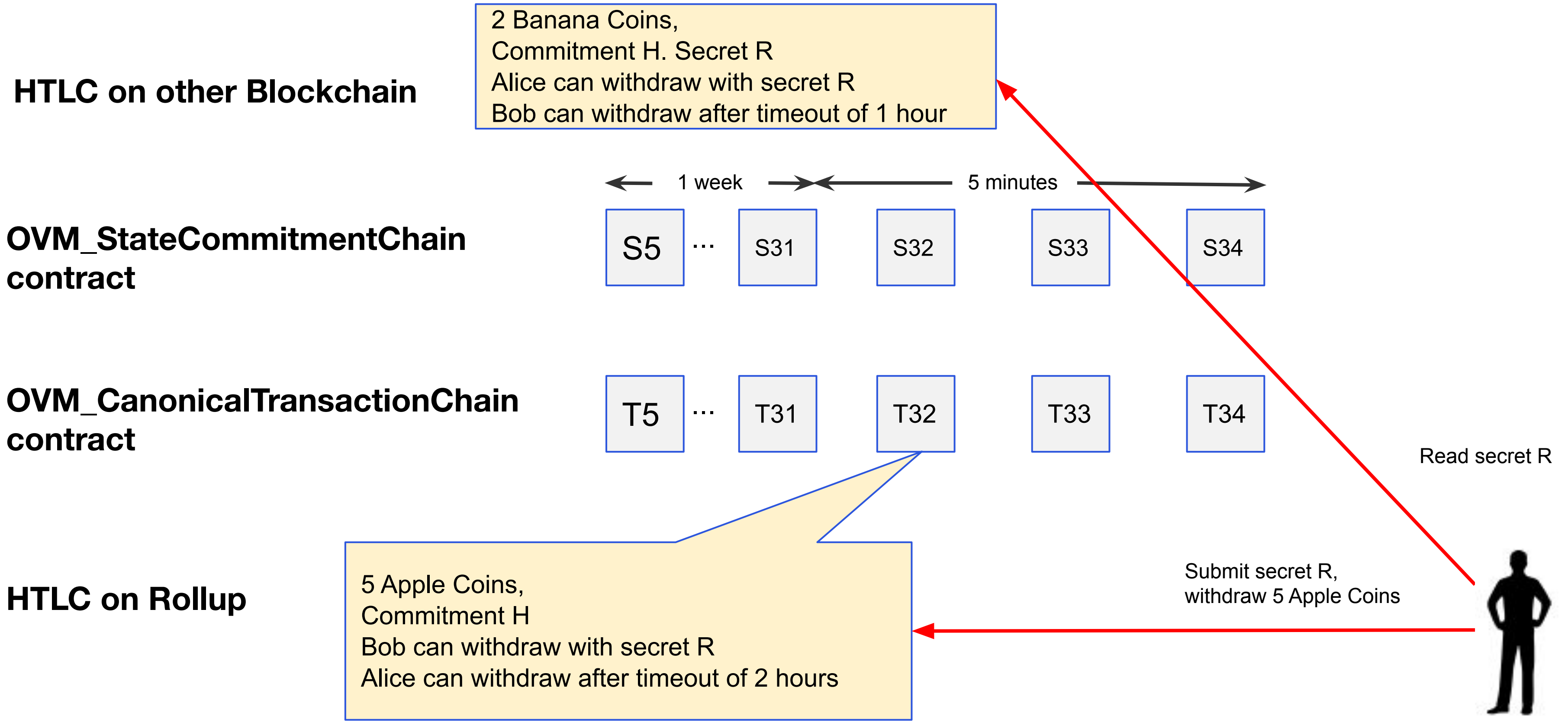
OVM_StateCommitmentChain contract

OVM_CanonicalTransactionChain contract

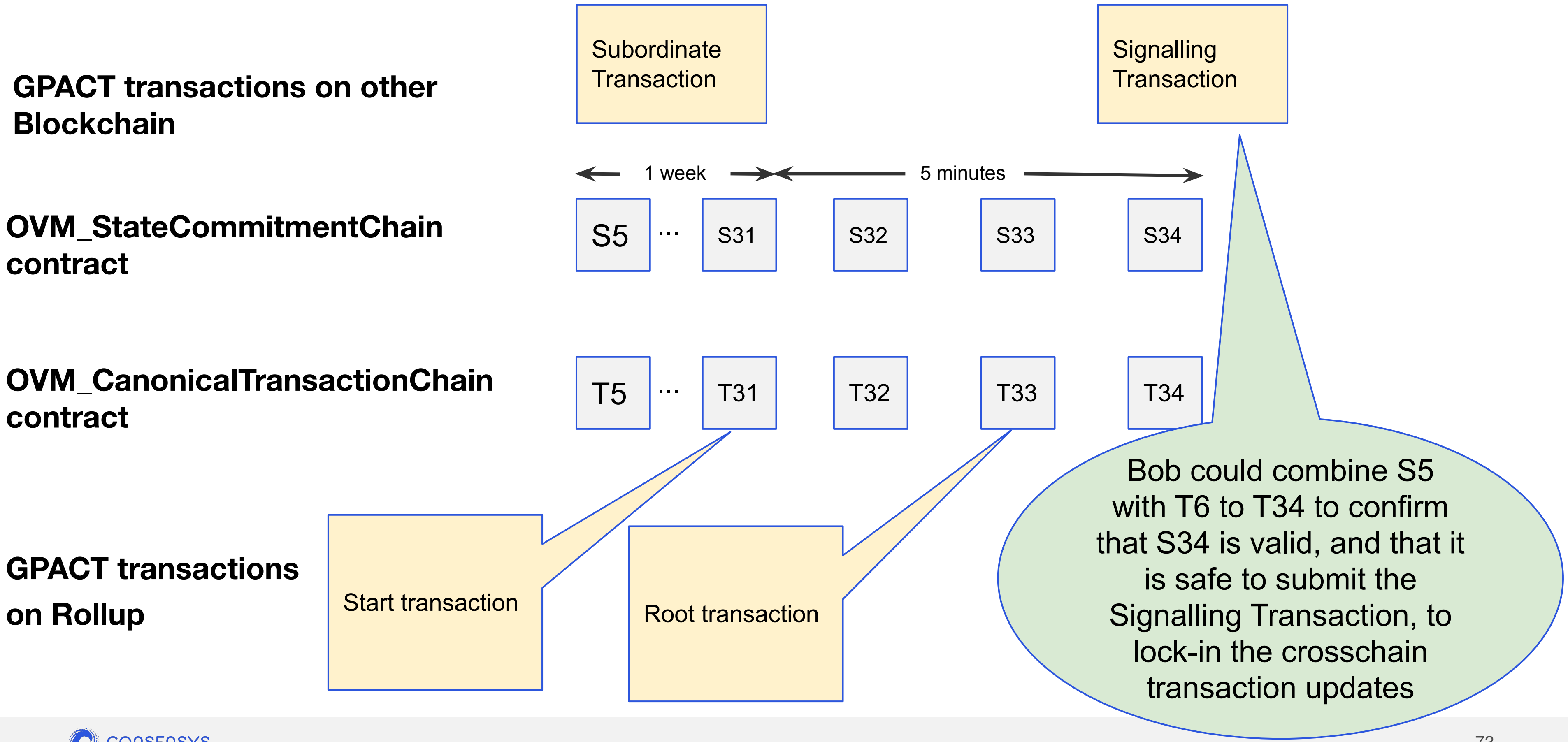
HTLC on Rollup

5 Apple Coins,
Commitment H
Bob can withdraw with secret R
Alice can withdraw after timeout of 2 hours

Optimism with HTLC


HTLC on other Blockchain
2 Banana Coins,
Commitment H. Secret R
Alice can withdraw with secret R
Bob can withdraw after timeout of 1 hour
1 week
5 minutes
OVM_StateCommitmentChain contract
S5
…
S31
S32
S33
S34
OVM_CanonicalTransactionChain contract
T5
…
T31
T32
T33
T34
Read secret R
HTLC on Rollup
5 Apple Coins,
Commitment H
Bob can withdraw with secret R
Alice can withdraw after timeout of 2 hours
Submit secret R,
withdraw 5 Apple Coins

Optimism with GPACT (atomic crosschain function calls)



GPACT transactions on other Blockchain
Subordinate Transaction
Signalling Transaction
1 week
5 minutes
OVM_StateCommitmentChain contract
S5
...
S31
S32
S33
S34
OVM_CanonicalTransactionChain contract
T5
...
T31
T32
T33
T34
GPACT transactions on Rollup
Start transaction
Root transaction
Bob could combine S5 with T6 to T34 to confirm that S34 is valid, and that it is safe to submit the Signalling Transaction, to lock-in the crosschain transaction updates

Crosschain Front Running

Contracts involved in crosschain transactions / messages need to be written with Front Running in mind.

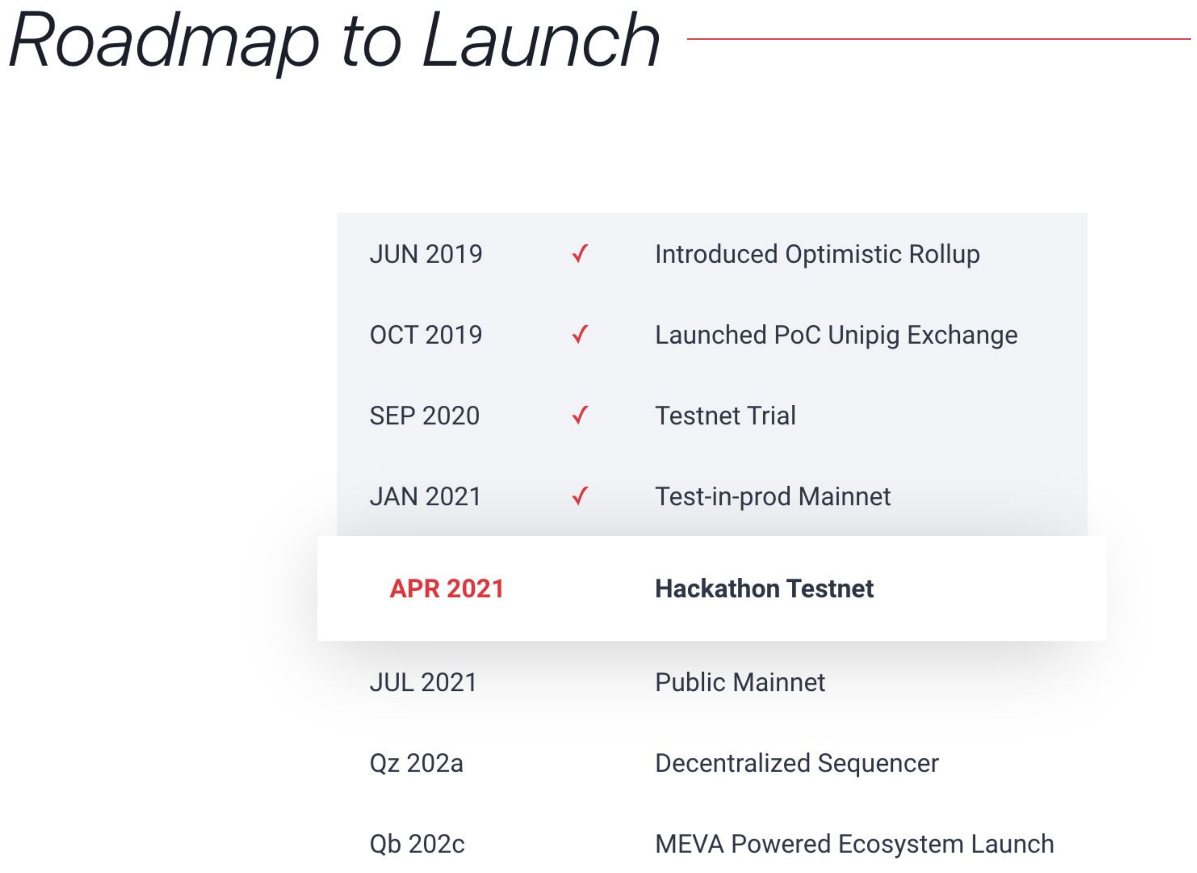
An example of what not to do:

- Auction application with function call: bid $2 more than the previous highest bidder.

The same rules to prevent single blockchain front running attacks apply:

- Have updates set absolute values, and not relative values.
- Have updates relative to a specified current value.

Dates



Roadmap to Launch

JUN 2019	✓	Introduced Optimistic Rollup
OCT 2019	✓	Launched PoC Unipig Exchange
SEP 2020	✓	Testnet Trial
JAN 2021	✓	Test-in-prod Mainnet
APR 2021		**Hackathon Testnet**
JUL 2021		Public Mainnet
Qz 202a		Decentralized Sequencer
Qb 202c		MEVA Powered Ecosystem Launch

Optimism Things to Consider

- Replacing opcode with function calls:
  - Amount of Gas used on Roll-up might be significantly more than on MainNet.
  - Code size will expand, possibly beyond the 24 kb limit.
- Will the transactions be that much cheaper than on MainNet?
  - Still storing compressed transaction and State Roots to MainNet for each transaction.
  - The cost of this storage has to be paid for.
- Submitting Fraud Proofs is going to use a LOT of gas.
- State reversion (but not transaction reversion):
  - Fraud proofs can be submitted a week later.
  - Say the Roll-up has 100 tps.
  - This means 100 x 60 x 60 x 24 x 7 states hashes could be reverted.
  - However, the transactions are still there.
  - 🤔 Other sequencers that built on top of the bad state would be slashed, as well as the bad sequencer. This means that fraud is likely to be detected much earlier than a week.

Optimism Things to Consider

Limits of Fraud Proofs:

- Prove invalid state transition.
- Transactions are still maintained.
- However, a transaction that was valid may become invalid.
    - Consider the simple example: transfer 1 Banana Token from account A to B.

What about:

- How many sequencers are there really? (the answer might be 1)
- Are the MainNet contracts upgradeable? (the answer might yes)
    - Need to think through the implications.

ARBITRUM

Arbitrum

- I have not had time to work through Arbitrum in detail.
- Arbitrum may be ready to go live before Optimism.
- Based on blog posts, my initial feel is that Optimism and Arbitrum have each been incorporating each others ideas into their platforms.
- According to https://medium.com/offchainlabs/whats-up-with-rollup-db8cd93b314e, a big difference is how fraud proof is done.
  - Optimism: single pass
  - Arbitrum: multi-pass
- However, Optimism would appear to have multiple passes to create proof...

If you come from the Arbitrum team &
would like to do a deep-dive as a meet-up talk, please contact me!

A Thought Bubble Idea

Thought Bubble Idea Raghavendra Ramesh & I had

Thought bubble - there might be all sorts of things wrong with this that we haven't thought of (if there is, could someone please tell us!)

For existing Optimistic Roll-ups:

- Less MainNet gas would be used if only a Merkle Root of State Roots for a batch was stored on MainNet.
- For Fraud Proofs, a Merkle Proof showing the previous State Root could be constructed.
- The transactions would still need to be available.
- The amount of gas used to execute a Fraud Proofs would increase.

This approach is essentially what is planned for Ethereum with Data Shards.

zkRollups

zkRollups

zk = Zero Knowledge

zkRollups = Rollups that use Zero Knowledge Proofs.

Use zk technology to prove that transactions were executed correctly.

zkRollups

zk = Zero Knowledge

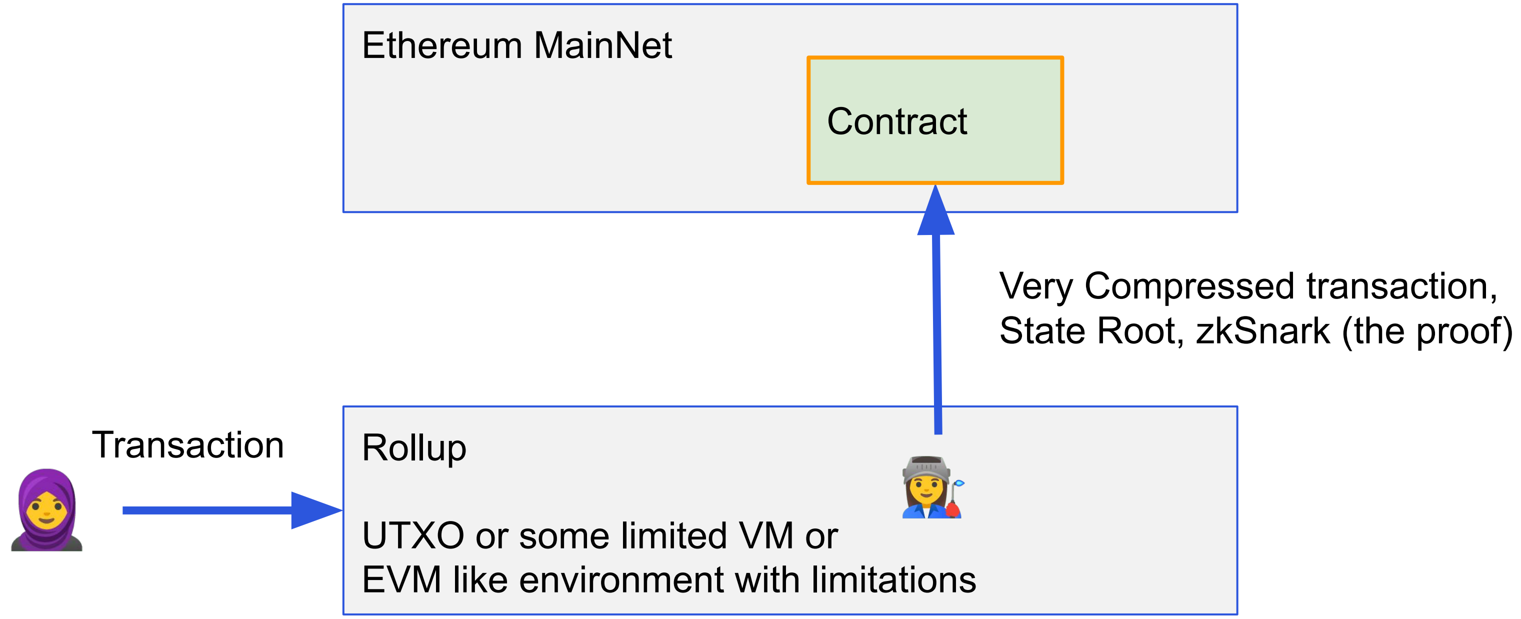
zkRollups = Rollups that use Zero Knowledge Proofs.

Use zk technology to prove that transactions were executed correctly.

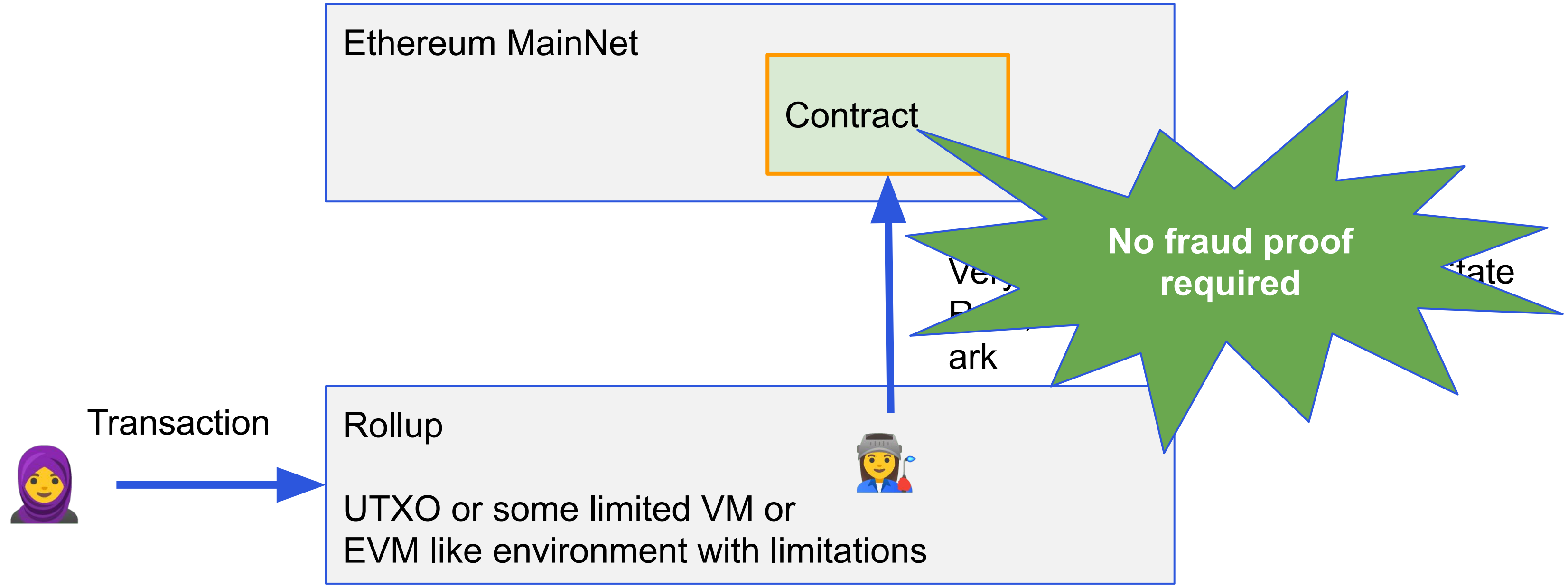
For a good introduction to zk, see:

https://z.cash/technology/zksnarks/ and https://github.com/matter-labs/awesome-zero-knowledge-proofs

zkRollups


Ethereum MainNet
Contract
Very Compressed transaction,
State Root, zkSnark (the proof)
Transaction
Rollup
UTXO or some limited VM or
EVM like environment with limitations

zkRollups


Ethereum MainNet
Contract
Ve
R
ark
tate
No fraud proof required
Transaction
Rollup
UTXO or some limited VM or
EVM like environment with limitations

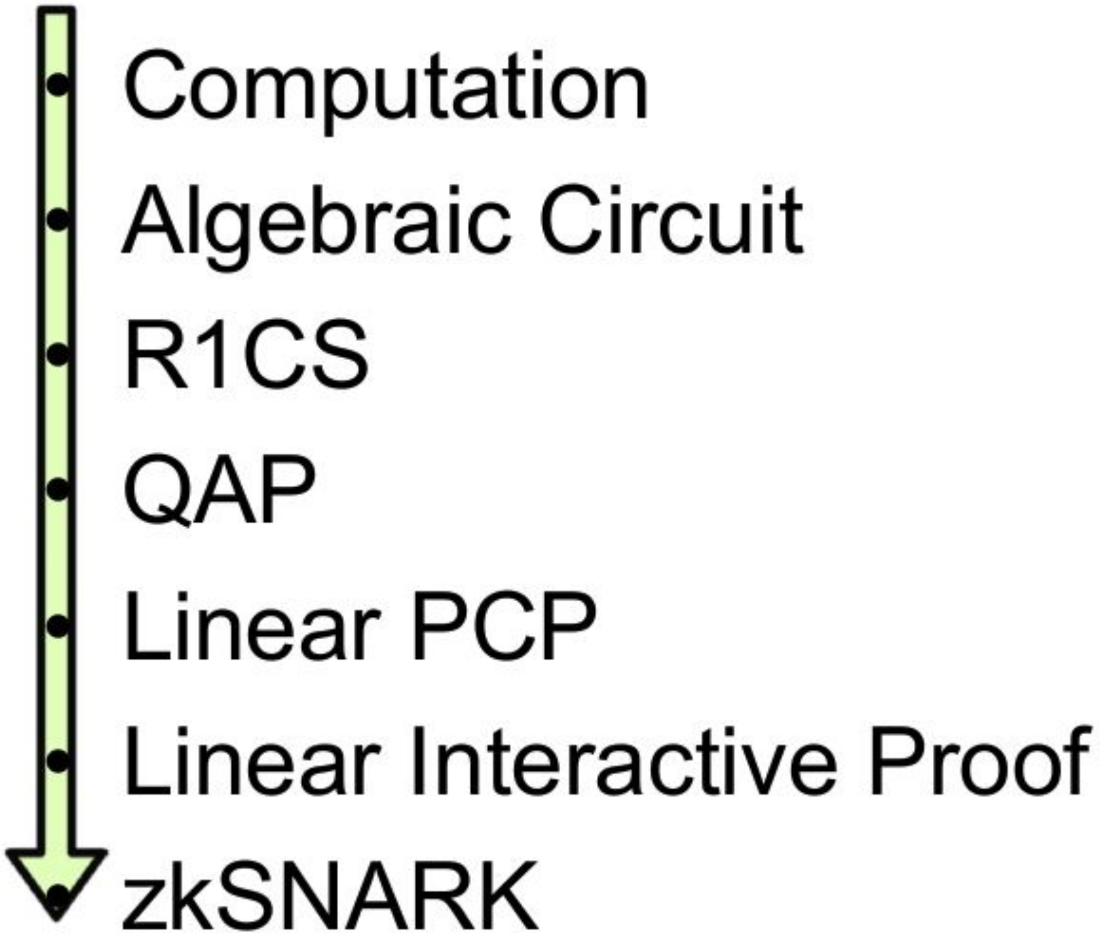
zk Schemes

There are lots of alternative zk scheme.

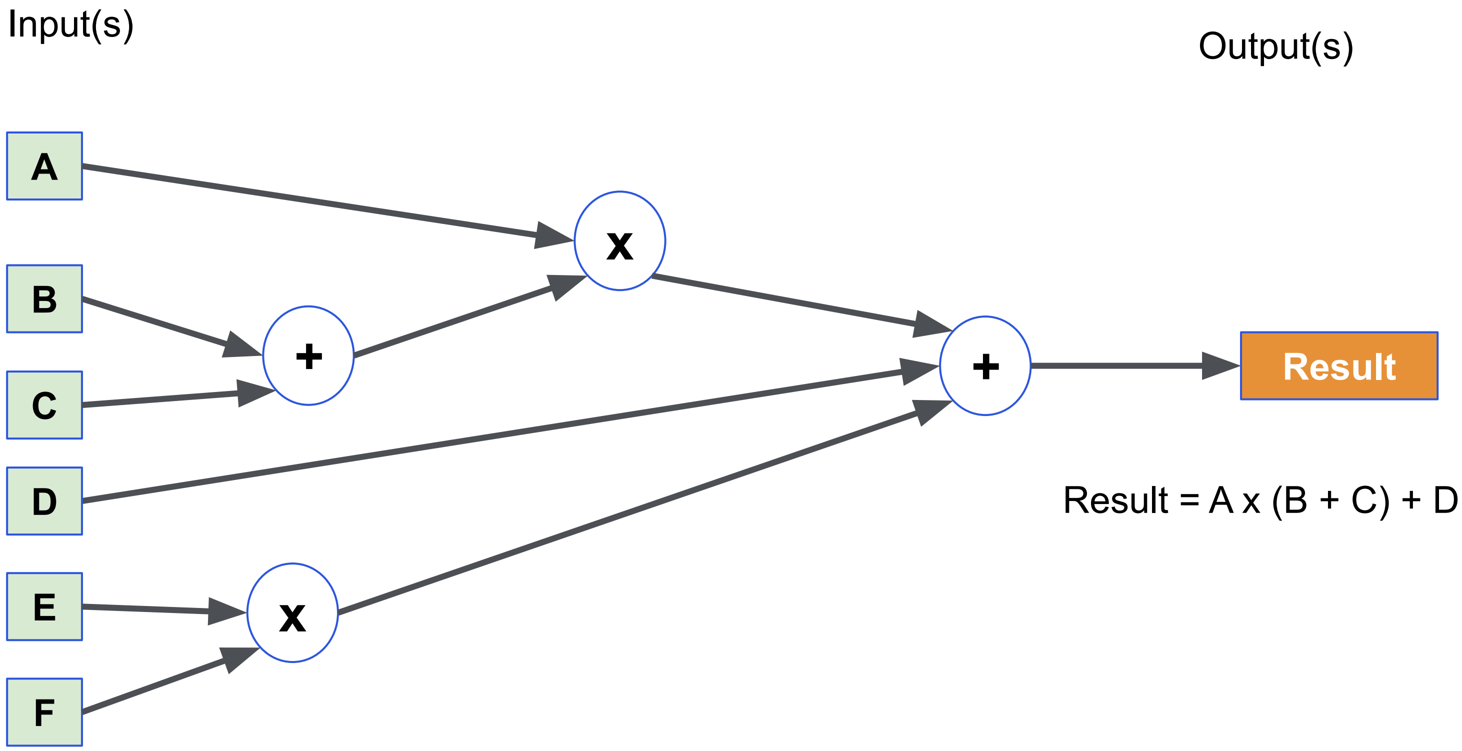
This talk is just going to work through zkSnarks for zkRollups.

Creating a zkSnark

Many steps involved in converting from a balance transfer or some program execution to a zkSnark.



Arithmetic Circuits



Result = A x (B + C) + D + E * F

zkSnark: Computation

- Proof creation:
  - Creating proofs takes a LOT of CPU.
  - **Typical batches of transactions takes in the order of minutes**.
  - Variable length of time: depends on computation (how big the circuit is) and the other algorithmic choices for the rest of the zkSnark.
    - For example: ZCash private transactions take 1 or 2 seconds on recent laptops / CPUs.
- Proof verification:
  - Fixed verification duration.
  - Involves two EC pairing operations: 105K gas for BN254 (aka BN128), assuming just two values.

Proof Size

Proof size depends on the scheme:

- Groth16 (zkSnark used by ZCash) compressed size is 128 bytes.
- PlonK (AZTEC, zkSync) size depends on instantiation of scheme: currently around 400 bytes.

zkSync

Zinc

Zinc programming language similar to Rust (according to MatterLabs) optimized for ZKP circuits.

- Security. It should be easy to write deterministic and secure applications. Conversely, it should be hard to write code to exploit some possible vulnerabilities found in other programming languages.
- Safety. The language must enforce the strictest semantics available, such as a strong static explicit type system.
- Efficiency. The code should compile to the most efficient circuit possible.
- Cost-exposition. Performance costs that cannot be optimized efficiently must be made explicit to the developers. An example is the requirement to explicitly specify the loop range with constants.
- Simplicity. Anyone familiar with C-like languages (Javascript, Java, Golang, C++, Rust, Solidity, Move) should be able to learn Zinc quickly and with minimum effort.
- Readability. The code in Zinc should be easily readable to anybody familiar with the C++ language family. There should be no counter-intuitive concepts.
- Minimalism. Less code is better. There should ideally be only one way to do something efficiently. Complexity should be reduced.
- Expressiveness. The language should be powerful enough to make building complex applications easy.

https://github.com/matter-labs/zinc

Zinc

- Turing incompleteness. **Unbounded looping and recursion are not permitted in Zinc**. This not only allows more efficient R1CS circuit construction but also makes formal verifiability about the call and stack safety easier and eliminates the gas computation problem inherent to Turing-complete smart contract platforms, such as EVM.

Zinc

Type system

We need to adapt the type system to be efficiently representable in **finite fields**, which are the basic building block of R1CS. The current type system mostly follows Rust, but some aspects are borrowed from smart contract languages. For example, Zinc provides integer types with 1-byte step sizes, like those in Solidity.

Ownership and borrowing

Memory management is very different in R1CS circuits compared to the von Neumann architecture. Also, since R1CS does not imply parallel programming patterns, a lot of elements of the Rust design would be unnecessary and redundant. Zinc has no ownership mechanism found in Rust **because all variables will be passed by value.** The borrowing mechanism is still being designed, but probably, only immutable references will be allowed in the future.

Loops and recursion

Zinc is a **Turing-incomplete** language, **as it does not allow recursion and variable loop indexes. Every loop range must be bounded with constant literals or expressions.**

zkSync 2.0

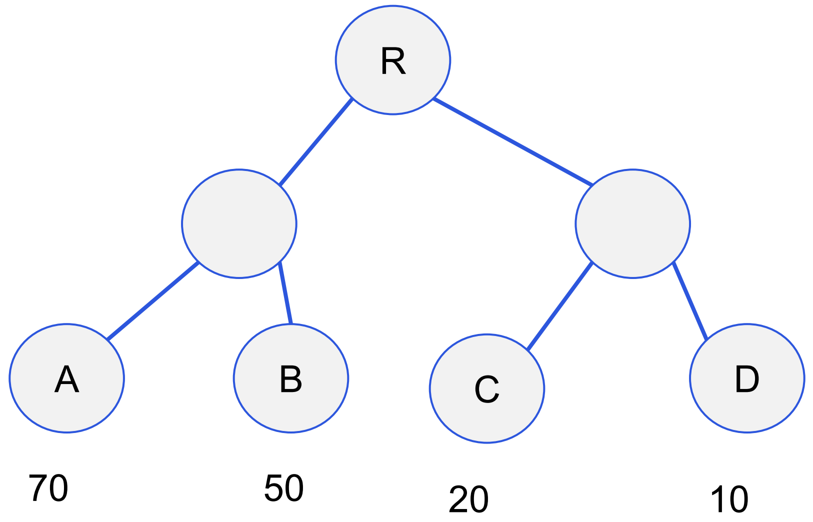
Application developers said that they wanted to use Solidity as is.

This should be kept in mind for people exploring custom VMs.

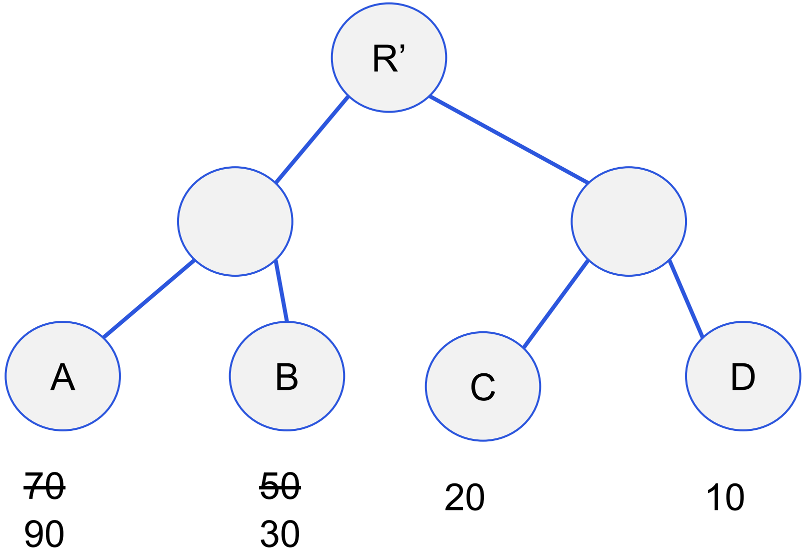
zkSync 2.0

- zkSync 2.0 is described really well here: https://www.youtube.com/watch?v=6wLSkpIHXM8
- Documentation on the zkSync website talks about earlier technology.
- The following slides borrows heavily from the youtube video.

State Transition Function



State Transition Function



R’ = STF(R, transactions, accounts, Merkle Proofs)

State Transition Function

Witness

R’ = STF(R, transactions, accounts, Merkle Proofs)

Recall:
Stateless Ethereum uses the same sort of thing.

Zero-knowledge proof of State Transition Function

R’ = STF(R, transactions, accounts, Merkle Proofs)

State transition function becomes the arithmetic circuit.

Zero-knowledge proof of State Transition Function

R’ = STF(R, transactions, accounts, Merkle Proofs)

input to circuit / proof

Arithmetic Circuit -> Prover -> Proof

Generic Computation using Tiny RAM

Imagine circuit is: if (op=ADD, acc=acc+input) else if op=MUL, acc=acc x input)

Step	Operation	Input	Accumulator
0	-	-	0
1	ADD	2	2
2	MUL	3	6
3	ADD	5	30

Generic Computation using Tiny RAM

- Need 1000x gates compared to hard wiring circuits.
- Can combine hard wired circuits for complex operations (keccak256) with general operations.

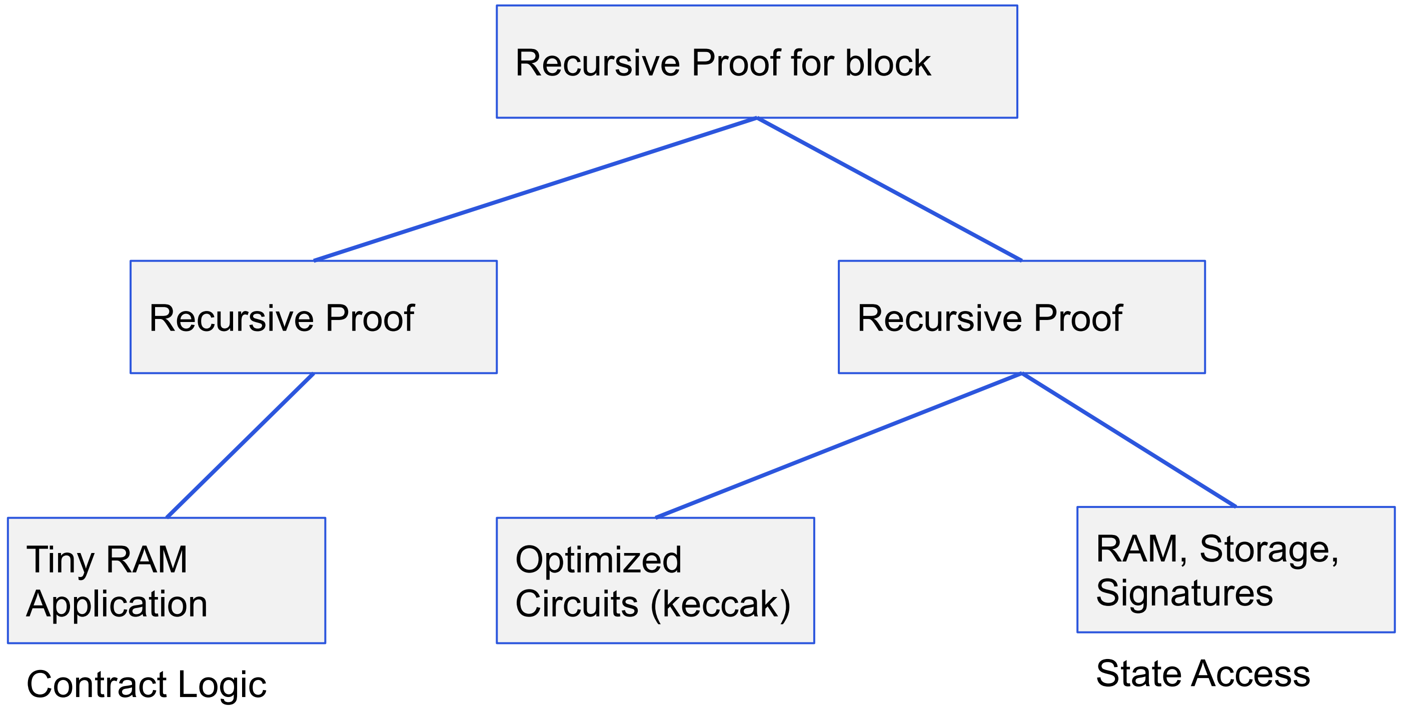
Recursive SNARKs

- Each prover depends on the circuit (the Solidity) that is being proved.
- This would lead to one prover per program…

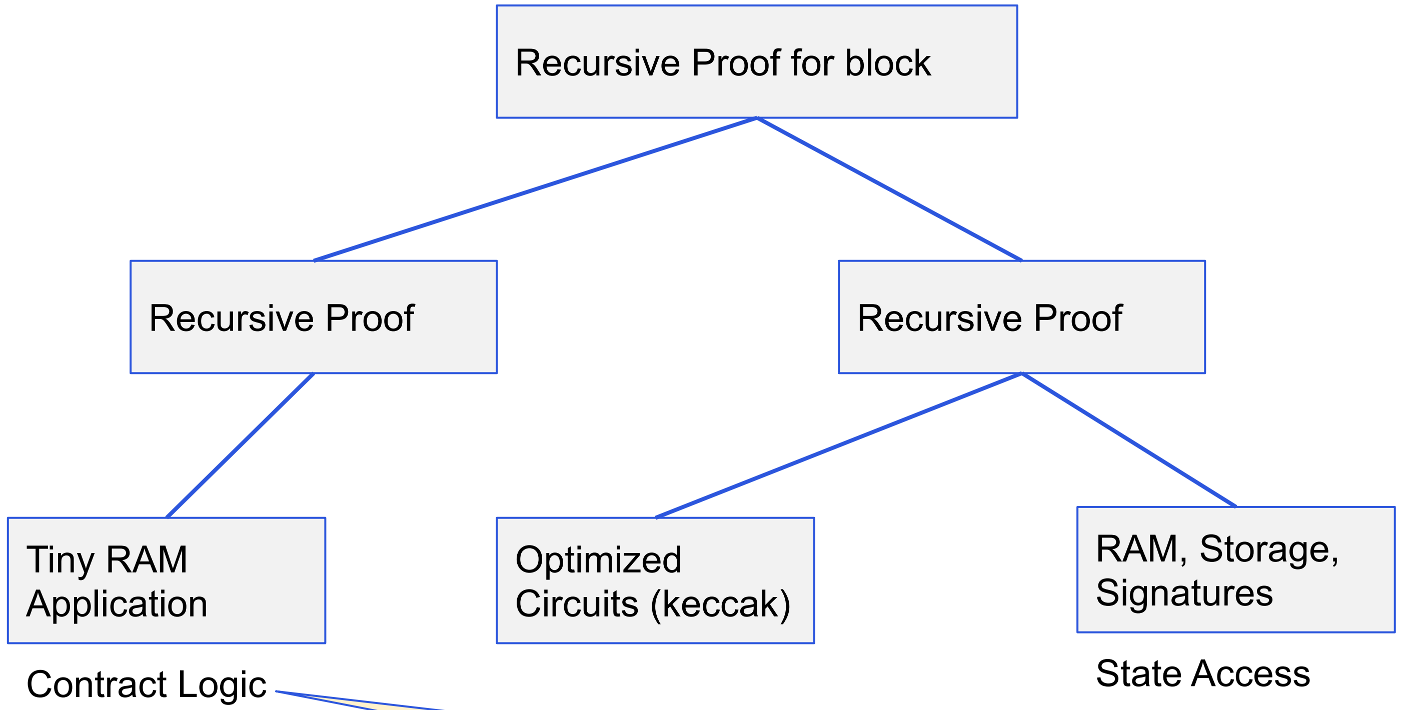
Recursive SNARKs

- Recursive SNARK: a circuit to verify other proofs

zkSync 2.0: Mixing Heterogeneous Circuits



zkSync 2.0: Mixing Heterogeneous Circuits



Only used for small parts of the overall code. Hence, the 1000x gate requirement is acceptable.

Solidity on zkSync

From: https://zksync.io/dev/contracts/#solidity

It ***will be*** possible to deploy most Solidity projects almost without modification. However, some features will likely be prohibited and should be omitted to keep the code compatible:

- ASM blocks with memory access
- Facilitating calculations via overflows
- ABI contract calls
- General cases of undefined behavior

I think this is:

address(contractAddr).call(abi.encode(...))

I think this is opcode such as:
now
blockhash
blocknumber

Gas

In zkSync 2.0, there is a different concept of gas. **Transaction prices will fluctuate depending on current L1 gas prices** (due to publishing calldata) and cost of ZKP generation. Smart contract calls will have a maximum number of zkEVM steps and storage write parameters.

From: https://medium.com/matter-labs/zksync-2-0-hello-ethereum-ca48588de179

What is stored on MainNet?

oldRootHash; newRootHash: proof; ((from1; to1; amount1; nonce1) (from2; to2; amount2; nonce2)...)

What is stored on MainNet?

(oldRootHash; newRootHash: proof; (from1; to1; amount1; nonce1) (from2; to2; amount2; nonce2)...)

Just a sec… no signatures?

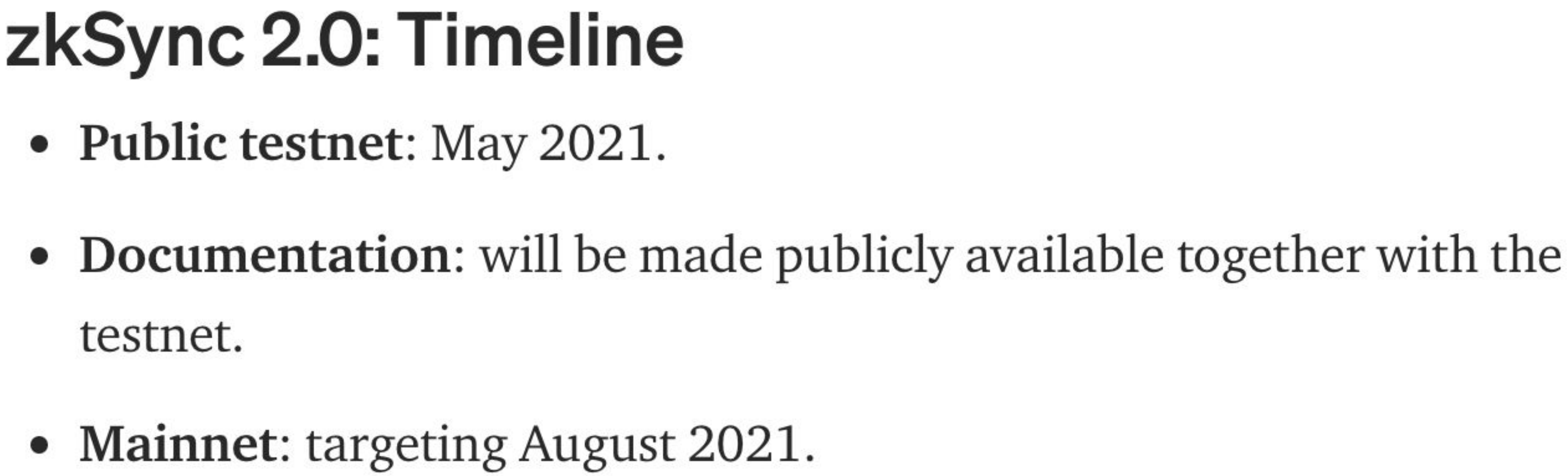
What is stored on MainNet?

- Transactions will be re-executed if someone (e.g. another operator) needs to recreate the state.
- But we don't need the signature, because when the operator sends the batch, it also sends the proof that it verified all signatures => the transactions were well signed.
- For example with 2 txs received by the operator creating the batch:

  (from1; to1; amount1; nonce1; sig1)

  (from2; to2; amount2; nonce2; sig2)
- The operator sends to the blockchain:
  - (oldRootHash; newRootHash: proof; (from1; to1; amount1; nonce1) (from2; to2; amount2; nonce2))
- When another operator wants to recreate the state by reading the blockchain it does not have sig1, sig2 but it knows that these signatures actually exist thanks to 'proof'.

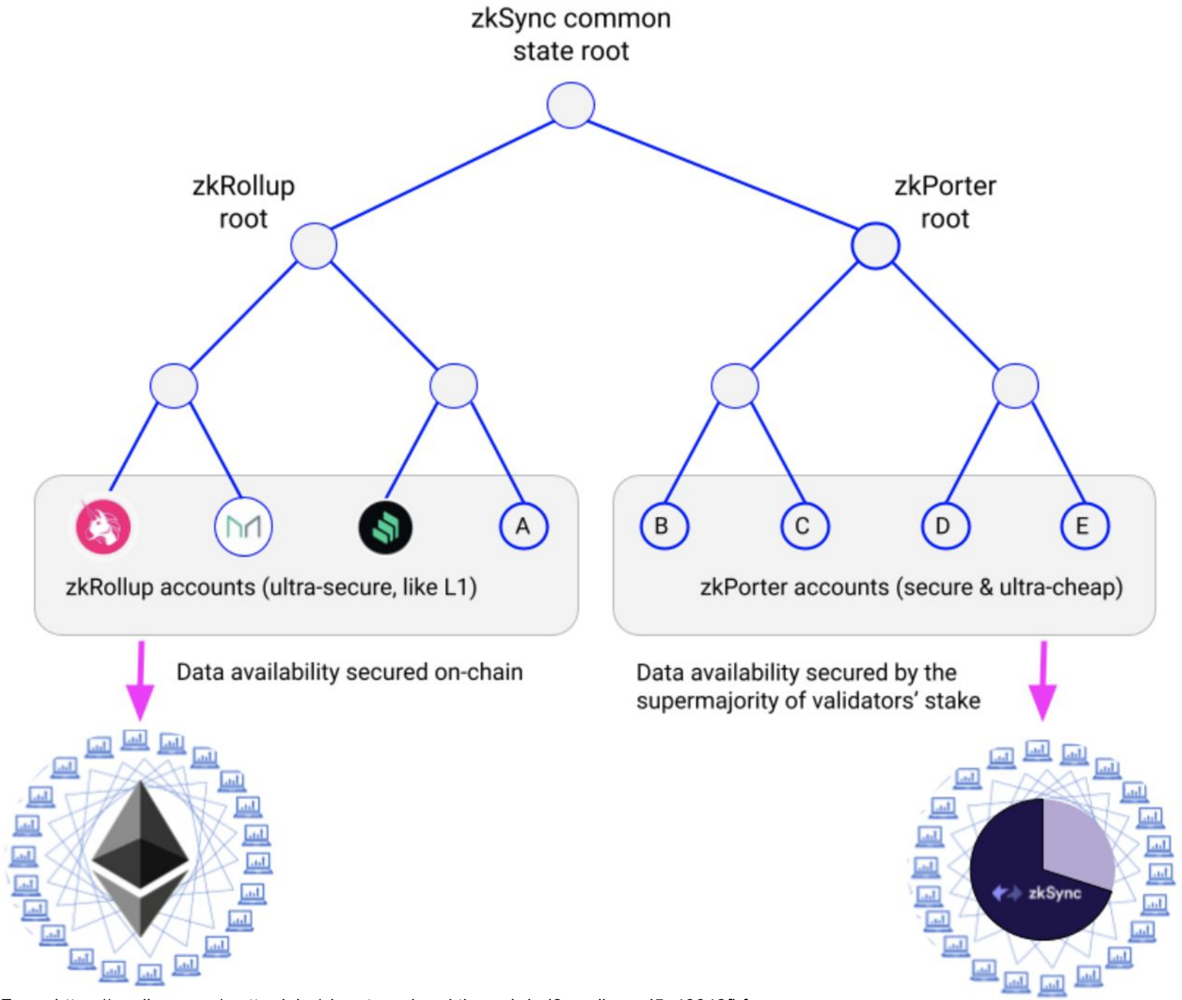
zkSync



zkSync 2.0: Timeline

- **Public testnet**: May 2021.
- **Documentation**: will be made publicly available together with the testnet.
- **Mainnet**: targeting August 2021.

zkPorter


zkSync common state root
zkRollup root
zkPorter root
A
B
C
D
E
zkRollup accounts (ultra-secure, like L1)
zkPorter accounts (secure & ultra-cheap)
Data availability secured on-chain
Data availability secured by the supermajority of validators' stake
zkSync

From: https://medium.com/matter-labs/zkporter-a-breakthrough-in-l2-scaling-ed5e48842fbf

STARKWARE

Starkware

Another major player in this space.

I have run out of time (and slides).

If you come from the Starkware team &
would like to do a deep-dive as a meet-up talk, please contact me!

CONSENSYS

ConsenSys

We are doing a lot of things in this space.

A lot of great stuff will be announced in due course.

Some open source contributions are public:

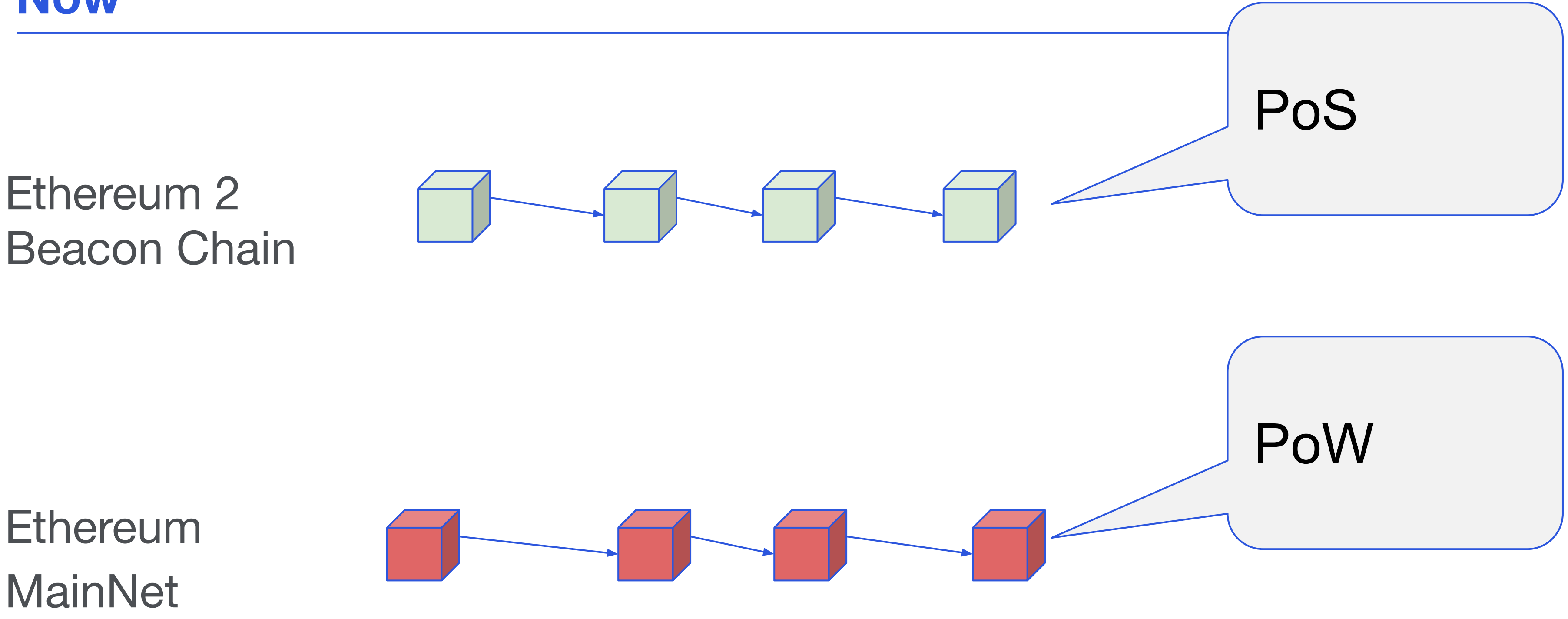
- gnark is a fast zk-SNARK library that offers a high-level API to design circuits.
  https://docs.gnark.consensys.net/en/0.4.0/

Some research results are public:

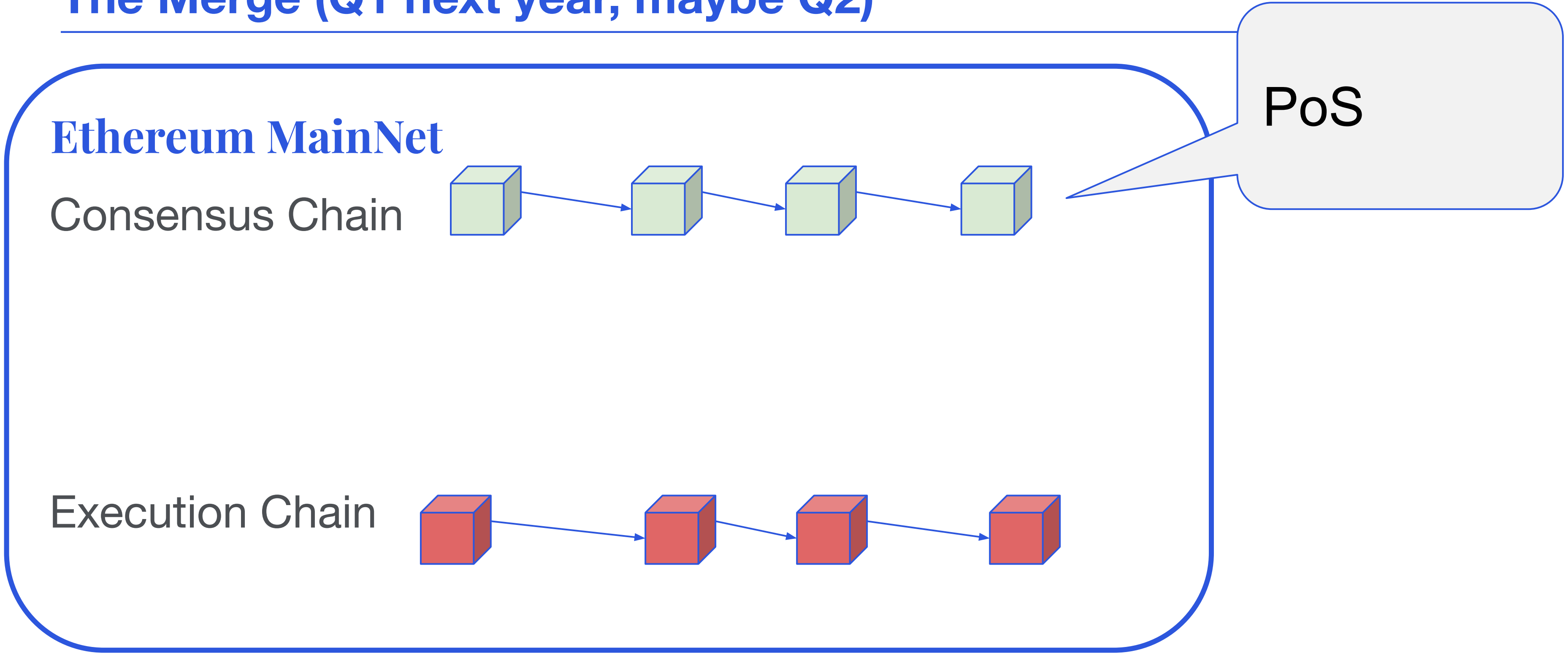
- Account-Based Anonymous Rollup: https://ethresear.ch/t/account-based-anonymous-rollup/6657
- Rollups on a data-sharded Ethereum 2: linking the data availability with the execution
  https://ethresear.ch/t/rollups-on-a-data-sharded-ethereum-2-linking-the-data-availability-with-the-execution/8237
- Sumcheck, GKR and Improving the Constraints per Hash Ratio in Snarks
  https://www.youtube.com/watch?v=uMuHyRChTQk

Ethereum (2) Data Shards and Execution Chain

Now

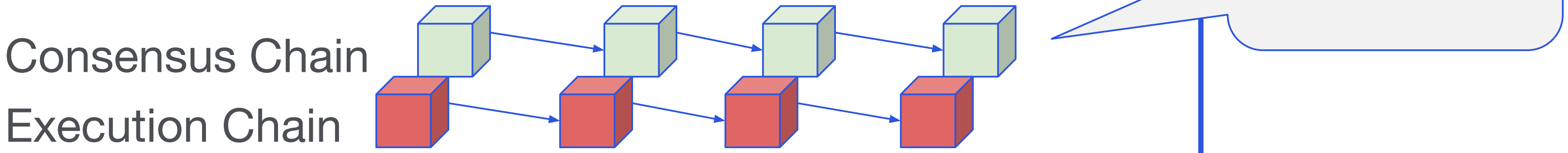


The Merge (Q1 next year, maybe Q2)



The Merge (Q1 next year, maybe Q2)

Ethereum MainNet



and then Data Shards

Ethereum MainNet

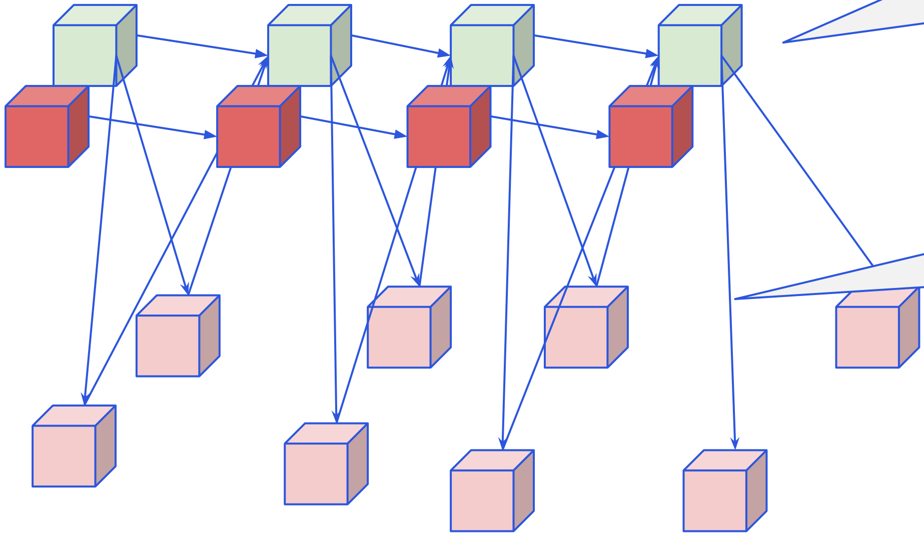
Consensus Chain

Execution Chain

Data Shard 0

Data Shard ...

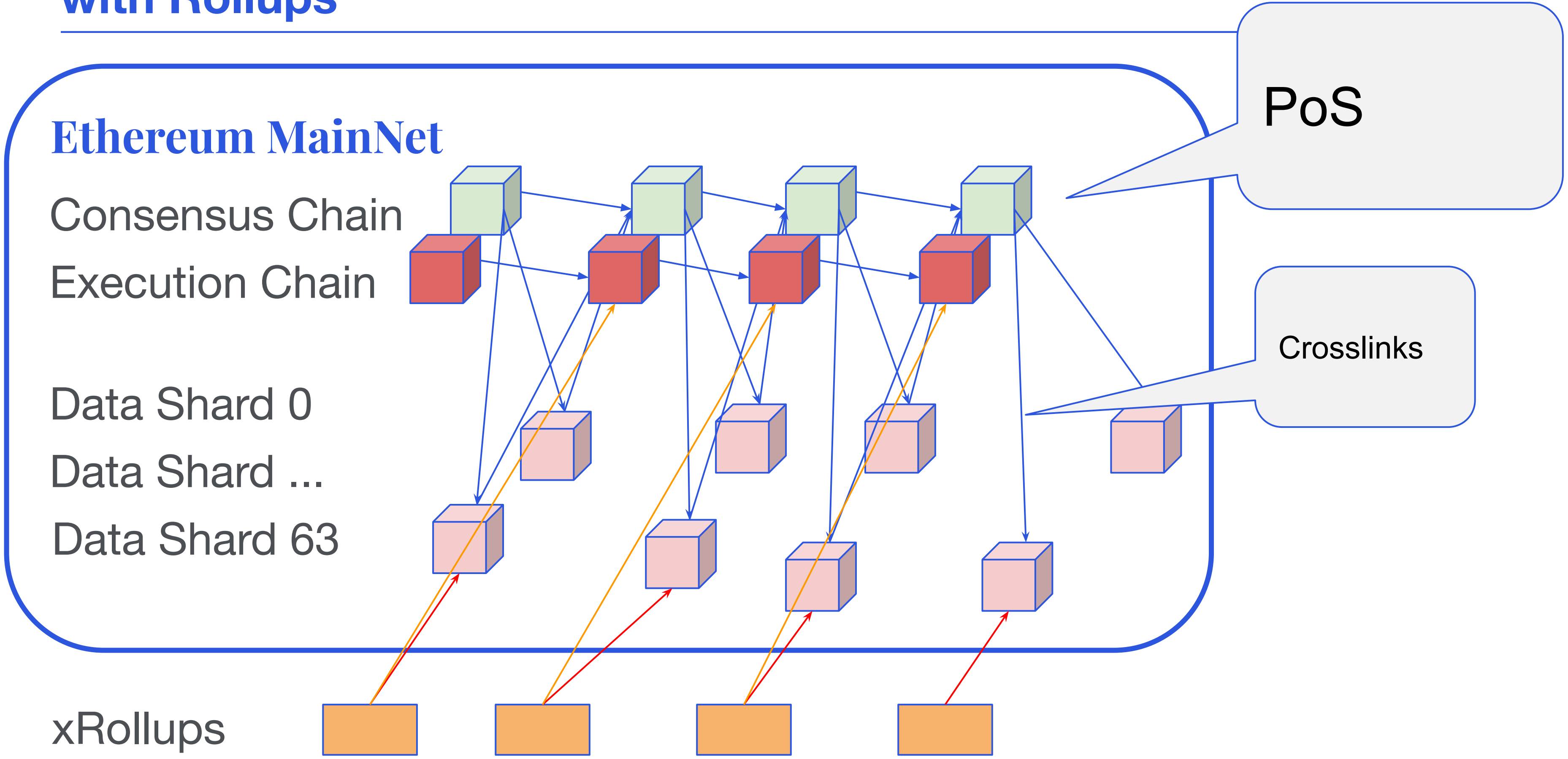
Data Shard 63



PoS

Crosslinks

with Rollups



Rollups and Ethereum (2)

From discussion with Nicolas Liochon:

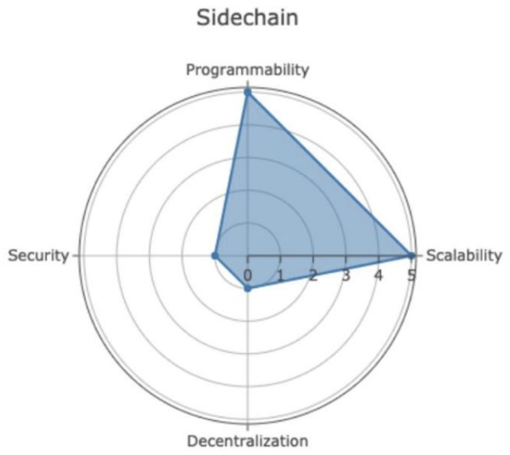
- Submit transactions to a Data Shard, updating state (adding a zkSnark for a state transition or a compressed transaction & state root)
- Submit transaction on Execution Chain, reference point / check point.
- Proofs on Execution Shard needs combination of reference point & proof with cross link

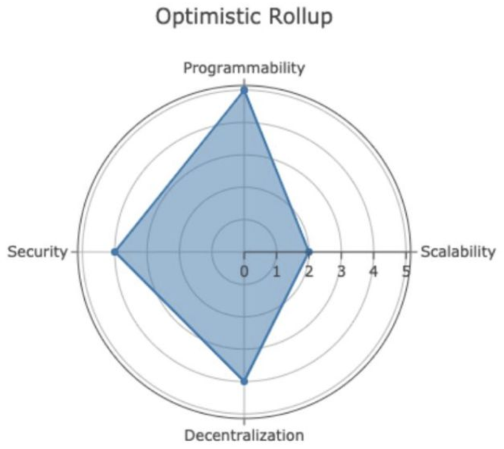
Comparison

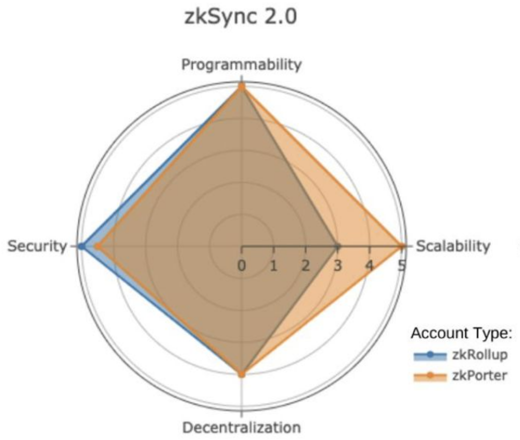
Comparison

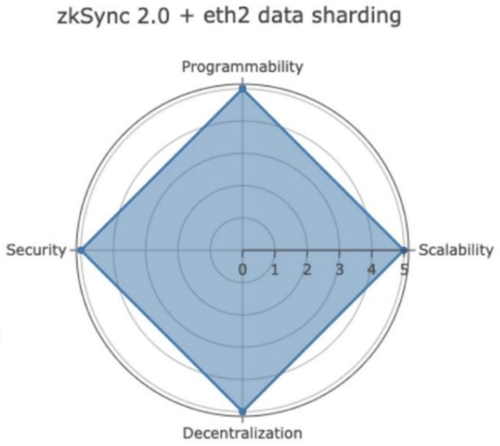
Technology	Consensus	EVM Compatibility	Withdrawal Latency	Security Model	Disputes	Data Availability
Sidechain	Separate	100%	Pin dispute window	Reputation / Trust	External: Court of Law	no
Optimistic Roll-up	Uses MainNet consensus	Mainly. Special compiler	Fraud Window	Cryptoeconmic	Fraud Proofs	yes
zkRollup	Uses MainNet consensus	Depends on system / limited	Immediate (once posted)	Cryptographic	zk	yes

Comparison by Matter Labs



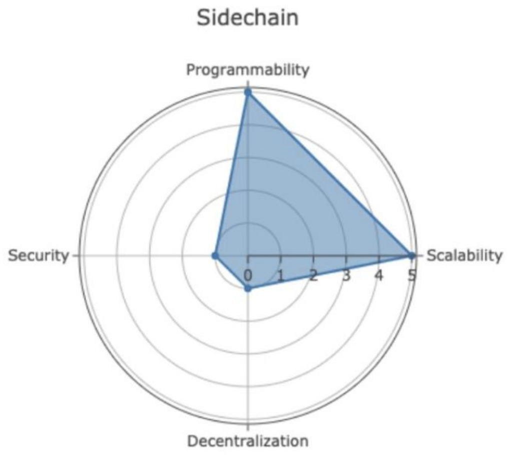


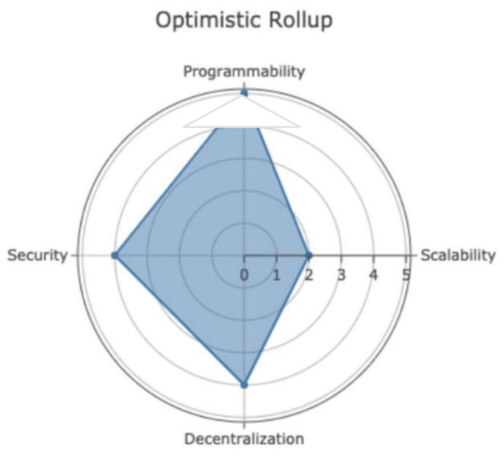


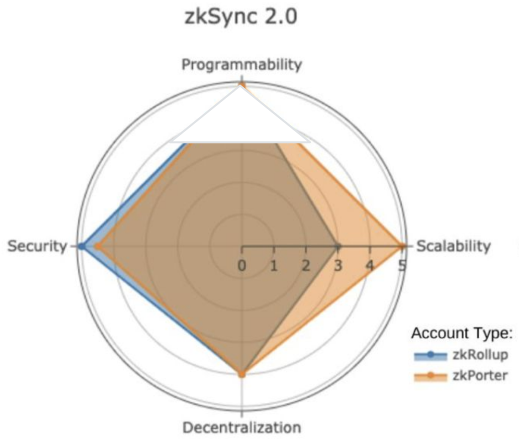


From: https://medium.com/matter-labs/zksync-2-0-hello-ethereum-ca48588de179

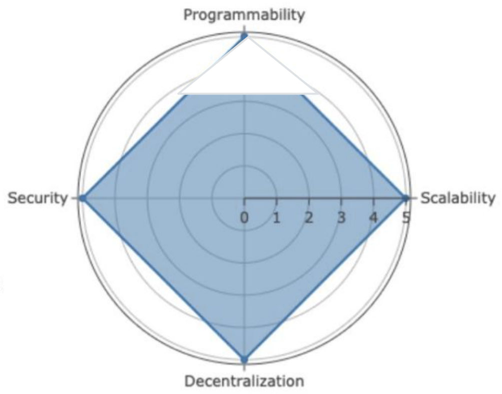
Peter's Comparison







zk or Optimistic +
Eth2 data sharding



Things to think through

- Fraud Proofs only work if at least one party submits a fraud proof.
- Optimistic VMs are complicated, though leverage in large part the EVM / current Ethereum Clients.
- zkRollups rely on newish cryptography. Tooling around creating a circuit appears to be super complicated.

Withdrawal Latency vs Finality

Withdrawal latency:

- Withdrawal latency is lower on zkRollups than Optimistic Rollups.

Finality:

- Transactions on Optimistic Rollups can be final faster than on zkRollups

Issues

Issues

Issues with the Layer 2 solutions are:

- Composability:
    - Ability to create an application consisting of a contract on one L2 that calls contracts on other L2s.
    - Compatibility of APIs for L2 solutions will also help.
    - Good crosschain communications will help solve this (think GPACT).
- On-boarding / Off-boarding friction:
    - Time and cost of moving assets between L1 and L2s, and from one L2 to another L2.
    - Means liquidity on L2s is, to an extent separated for liquidity on other L2s and on MainNet.
    - Good crosschain communications will help solve this (think GPACT).

Another Thought Bubble

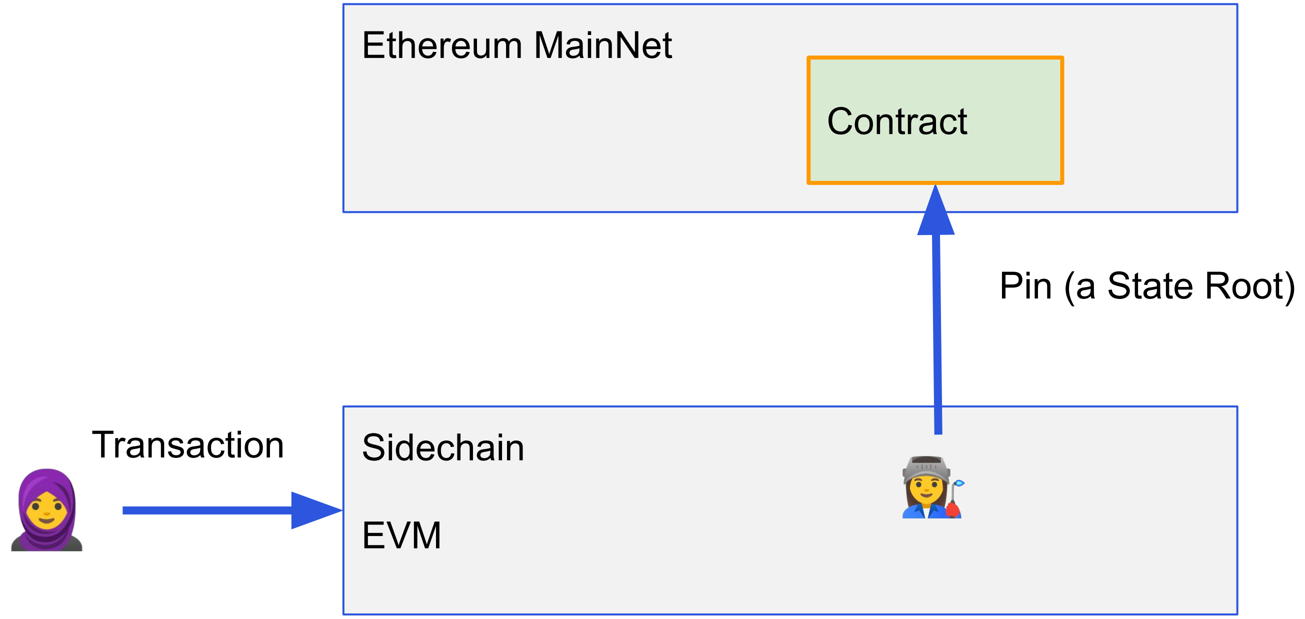
Idea from Nicolas Liochon

Combine Optimistic and zk:

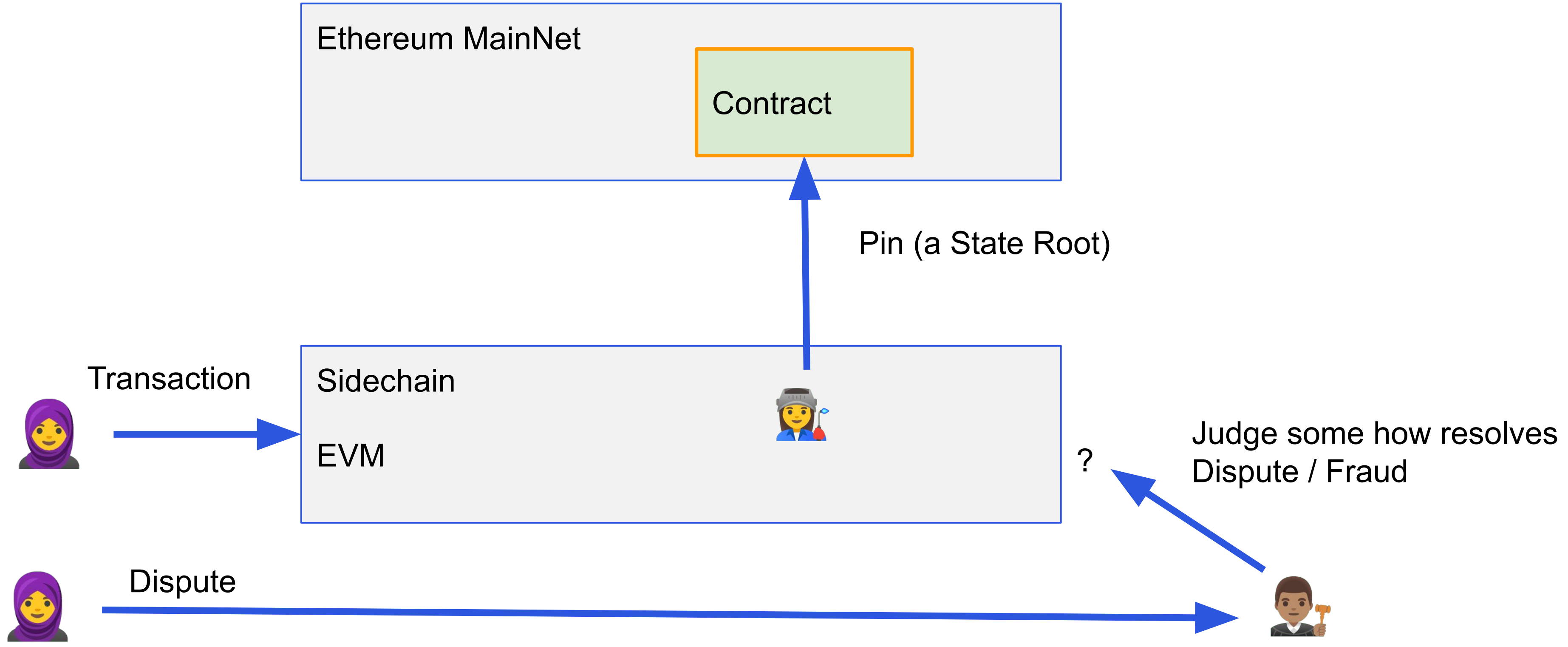
- Optimistic to do complicated for execution.
- zk to remove the signature, and thus have more compressed transactions.

Summary

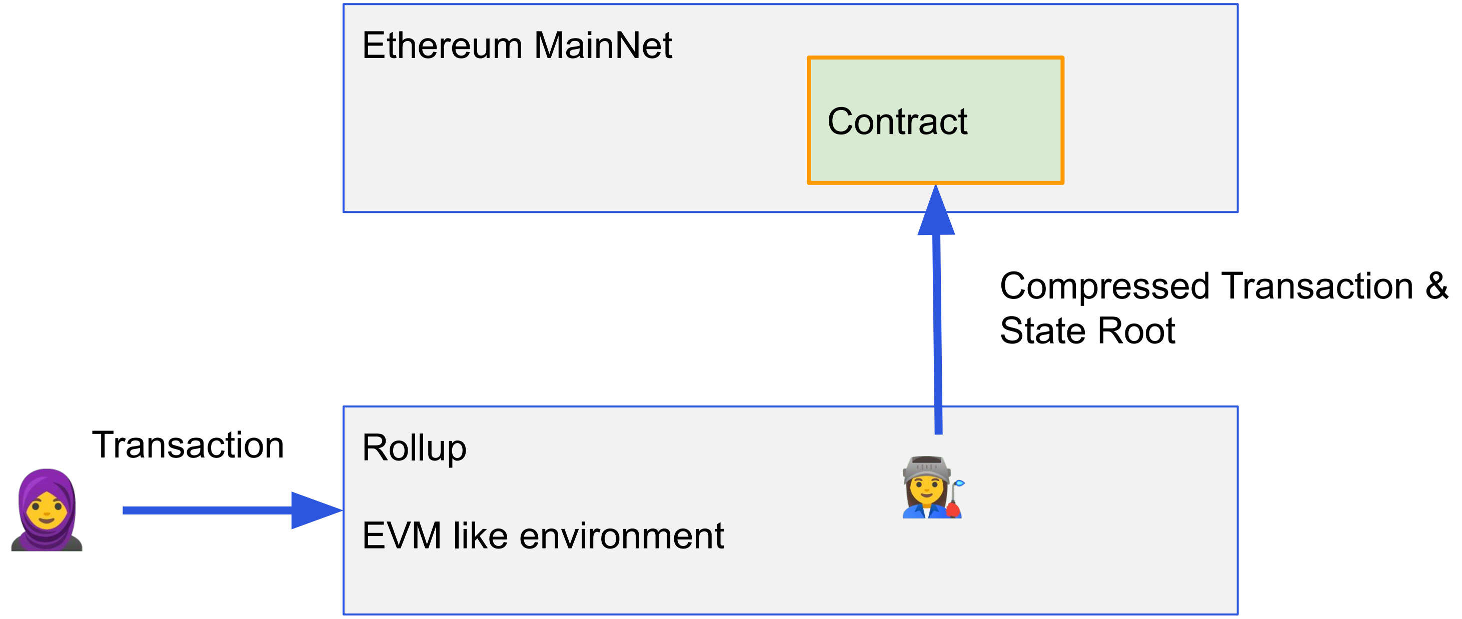
Sidechains


Ethereum MainNet
Contract
Pin (a State Root)
Transaction
Sidechain
EVM

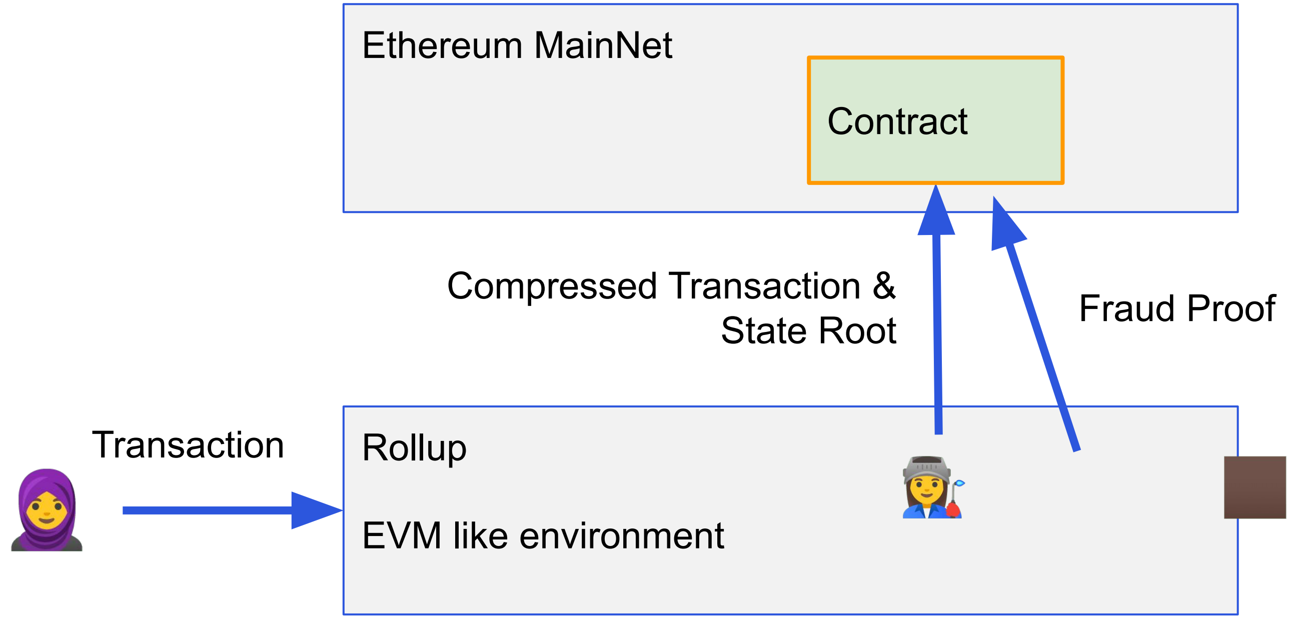
Sidechains


Ethereum MainNet
Contract
Pin (a State Root)
Transaction
Sidechain
EVM
?
Judge some how resolves
Dispute / Fraud
Dispute

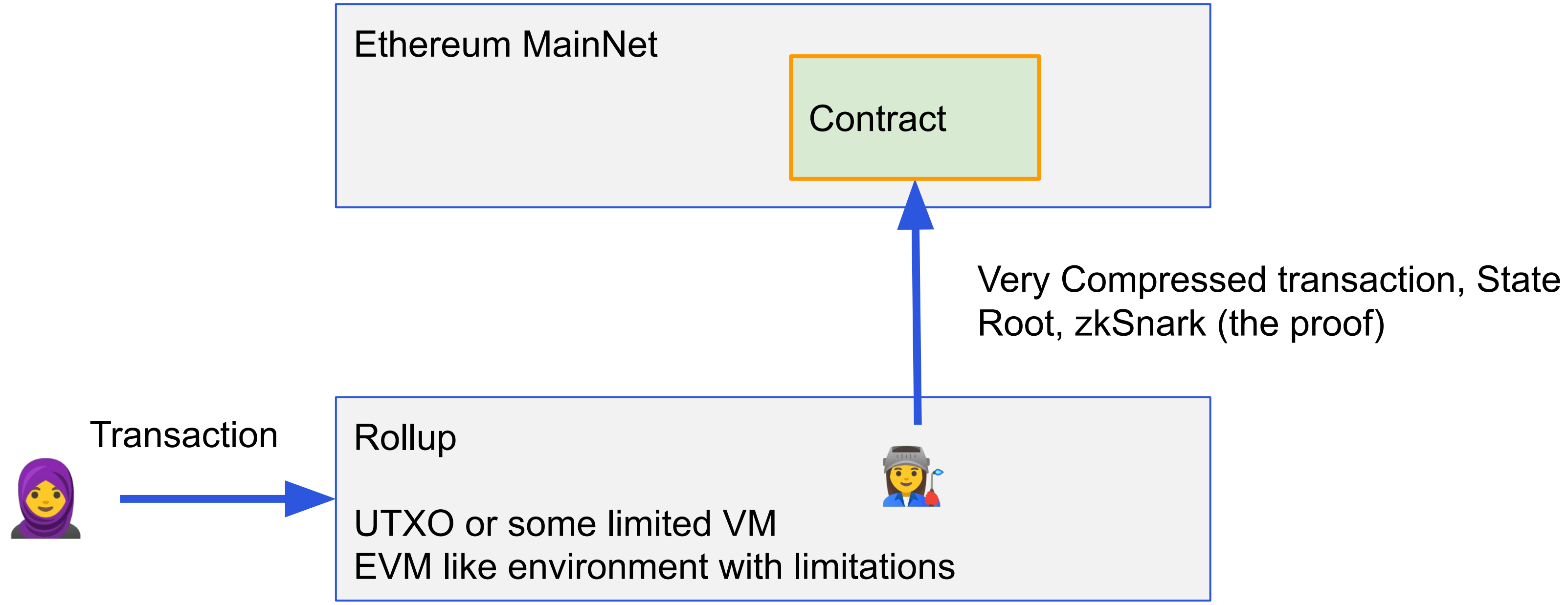
Optimistic Rollups



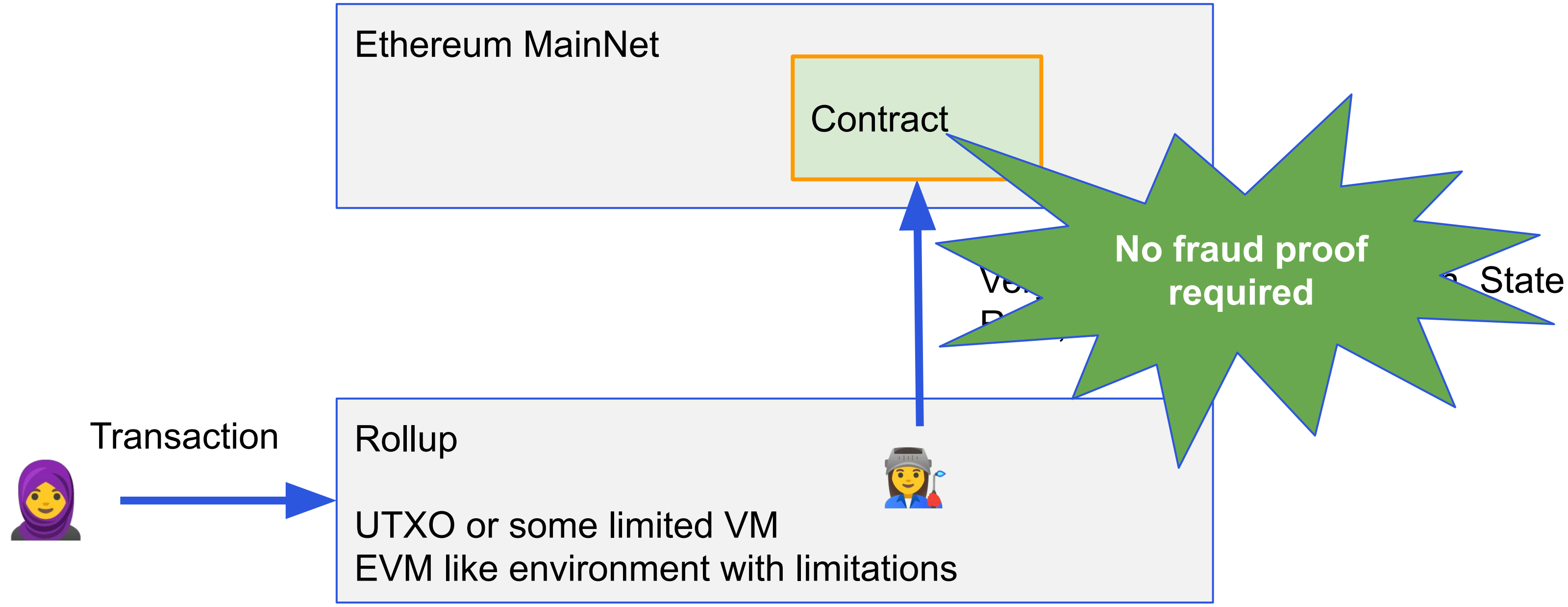
Optimistic Rollups



zkRollups


Ethereum MainNet
Contract
Very Compressed transaction, State Root, zkSnark (the proof)
Transaction
Rollup
UTXO or some limited VM
EVM like environment with limitations

zkRollups


Ethereum MainNet
Contract
No fraud proof required
Ve
State
Transaction
Rollup
UTXO or some limited VM
EVM like environment with limitations

Lots of Links!

https://vitalik.ca/general/2021/01/05/rollup.html

https://ethereum.org/en/developers/docs/scaling/layer-2-rollups/

Fuel

https://docs.fuel.sh/v1.1.0/Concepts/Fundamentals/Fuel%20Overview.html

Optimism

https://community.optimism.io/docs/

https://research.paradigm.xyz/optimism

Arbitrium

https://medium.com/offchainlabs/how-arbitrum-rollup-works-39788e1ed73f

zkSync

https://www.youtube.com/watch?v=6wLSkpIHXM8

https://medium.com/matter-labs/zksync-2-0-hello-ethereum-ca48588de179

https://zksync.io/dev/#overview

https://zksync.io/dev/contracts/#solidity

https://youtu.be/MstuXYewtXs

Reviewers

Thank you to the people who have reviewed early versions of these slides and helped with suggestions and ideas:

- Raghavendra Ramesh
- Ben Edginton
- John Adler
- Nicolas Liochon
- Gautam Botrel
- Mikhail Kalinin

Future Talks

June 23: Regulation of Central Bank Digital Currencies

- Simeon Lawson
- 12:30 pm Brisbane

July 6: Supply Chain on Blockchain Conference 2021: https://scobc.net/

- Free registration: https://www.eventbrite.com.au/e/supply-chain-on-blockchain-2021-tickets-150660855675

July 7: Tax Implications of Crypto & NFTs

- Felicity Deane
- 12:30 pm Brisbane

July 21: Waku and Ethereum Messaging

- Franck Royer

September 15: Using Bonsai Tries to improve Ethereum's performance

- Karim Taam and Gary Schulte
- 3pm Brisbane