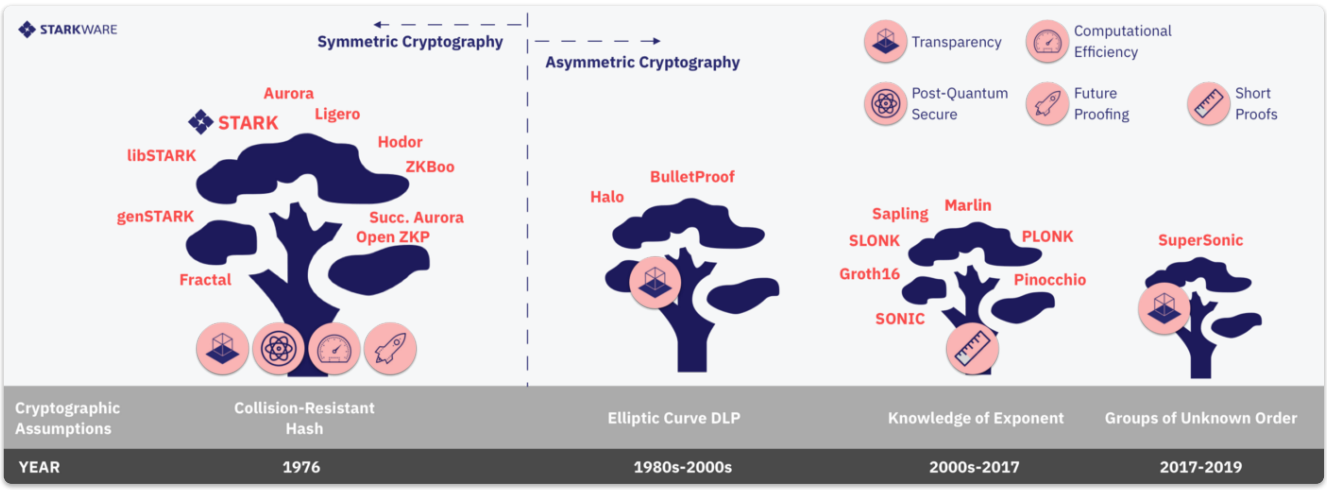
Lesson 3

Lesson 2 Review

General zkSNARK Process

1. Create a program that will test knowledge of the witness
2. Trusted setup, to create prover and verifier keys
3. Turn this program into an arithmetic circuit
4. From the arithmetic circuit create a R1CS
5. Further transformations are done to create the proof.

Real Life ZKP choices



SNARKS / STARKS are general purpose systems for creating proofs.

We don't always need SNARKS or STARKS other techniques may be more efficient for our use case.

Other useful techniques

- Blind signatures
- Accumulators
- Pedersen commitments
- Sigma protocols

Advantages of SNARKS

- Small proof size
- Fast verification
- Generic approach

Drawbacks to SNARKS

- Require a trusted setup
- Very long proving keys
- Proof generation is impractical on constrained devices
- Strong security assumptions haven't been well tested.
- Theoretical background is difficult to understand

Advantages of Sigma Protocols

- Very economical for small circuits
- Do not require a trusted setup
- Security assumptions are weak and are well understood
- Maths is fairly easy to understand

Disadvantages of Sigma Protocols

- Sigma protocols do not scale well, proof size, proof generation and verification size are linear in the complexity of the statement
- Cannot easily handle generic computation, they are better suited to algebraic constructions.

See the excellent blog post by Alex Pinto

Zokrates - a toolbox for zkSNARKs on Ethereum.

See repo
Zokrates was the first project to allow (easy) creation of proofs on Ethereum

ZoKrates helps you use verifiable computation in your DApp, from the specification of your program in a high level language to generating proofs of computation to verifying those proofs in Solidity.
Documentation : Zokrates

A preview of the process flow in Zokrates

We can see this workflow in Zokrates :

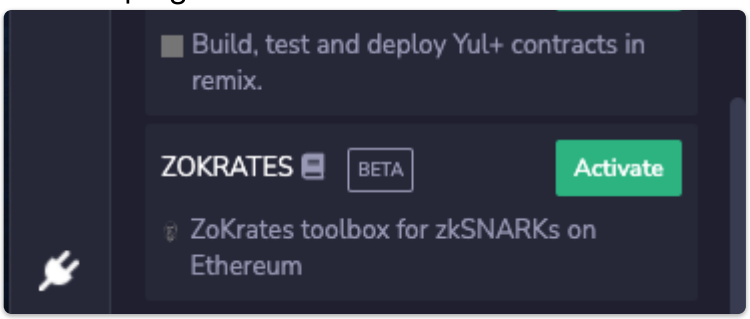
-The Creator writes and compiles a program in the Zokrates DSL

- The Creator / Prover generates a trusted setup for the compiled program
- The Prover computes a witness for the compiled program
- The Prover generates a proof - Using the proving key, she generates a proof for a computation of the compiled program
- The Creator / Prover exports a Verifier - Using the verifying key she generates a Solidity contract which contains the generated verification key and a public function to verify a solution to the compiled program

We can use Zokrates in Remix , it is also available as a program on Linux / Mac / Windows, or a docker container

Zokrates in Remix

Use the plugins menu to find and activate Zokrates

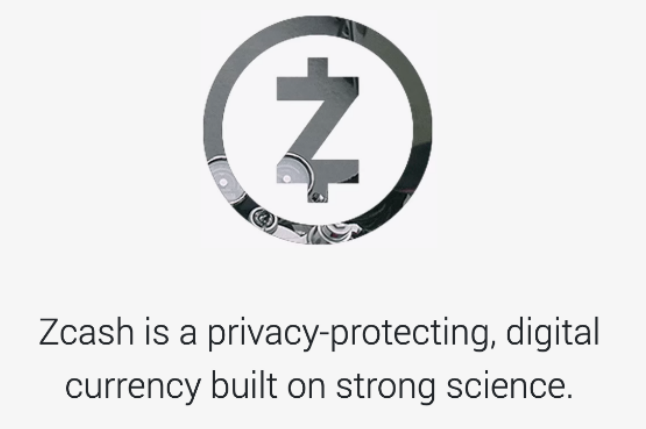


The Zokrates DSL is more limited than say Solidity, see the documentation

---

ZKP Use Cases

Privacy preserving cryptocurrencies



Also Nightfall , ZKDai

We will cover ZCash in more detail in a later lesson.

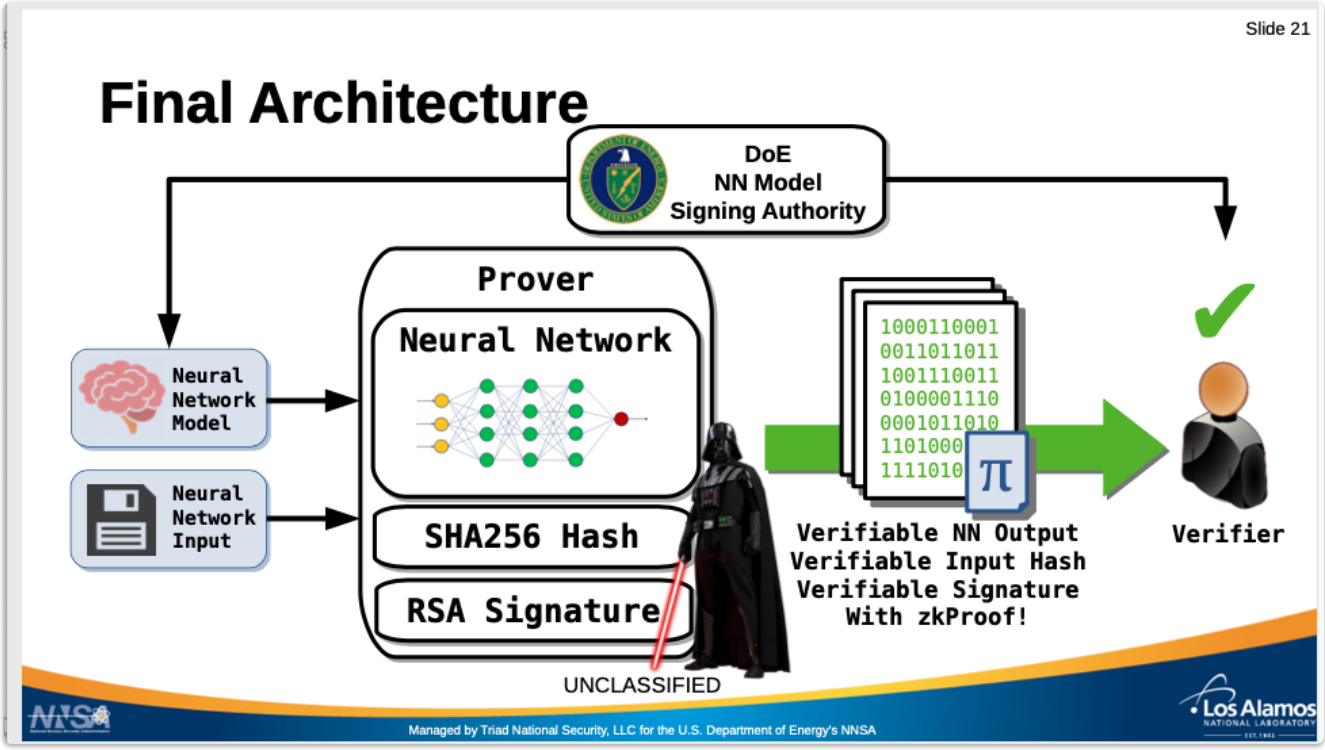
Nuclear Treaty Verification

Los Alamos NATIONAL LABORATORY EST. 1943

LA-UR-20-20260

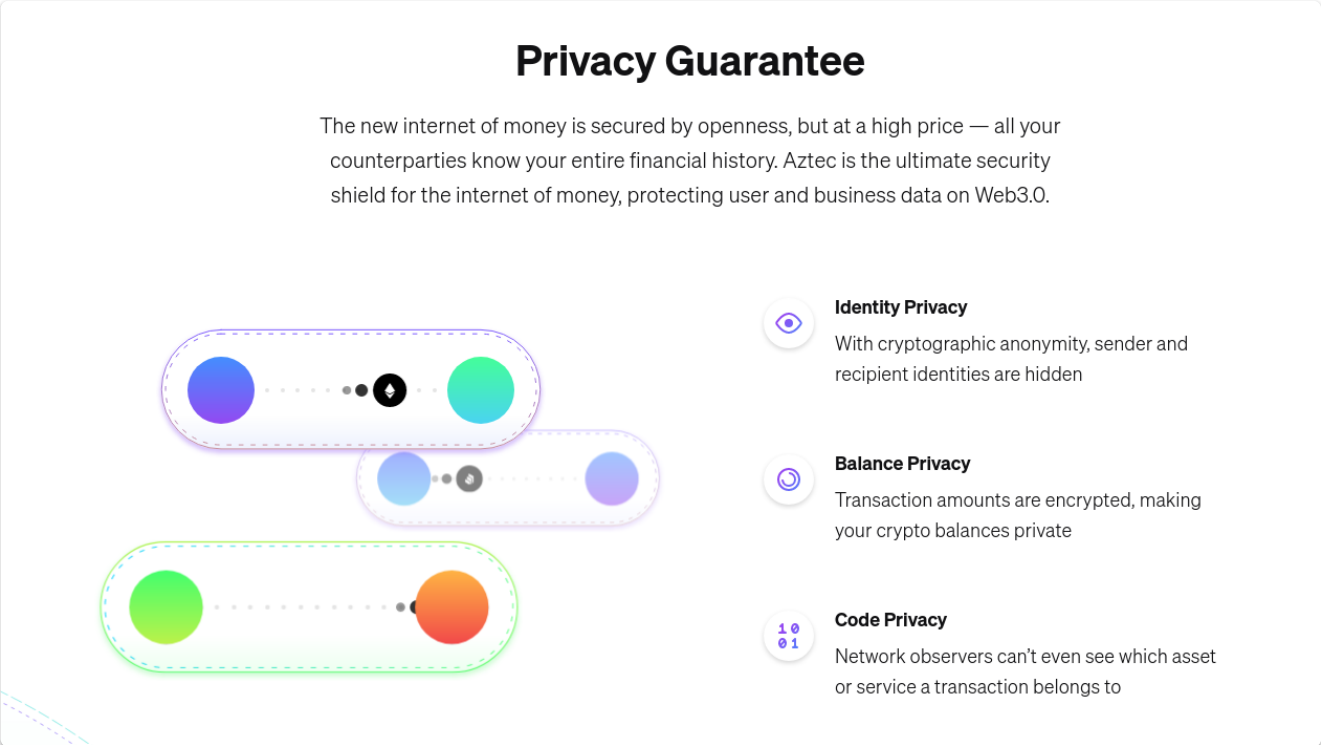
Approved for public release; distribution is unlimited.

Title:	SNNzkSNARK An Efficient Design and Implementation of a Secure Neural Network Verification System Using zkSNARKs
Author(s):	DeStefano, Zachary Louis



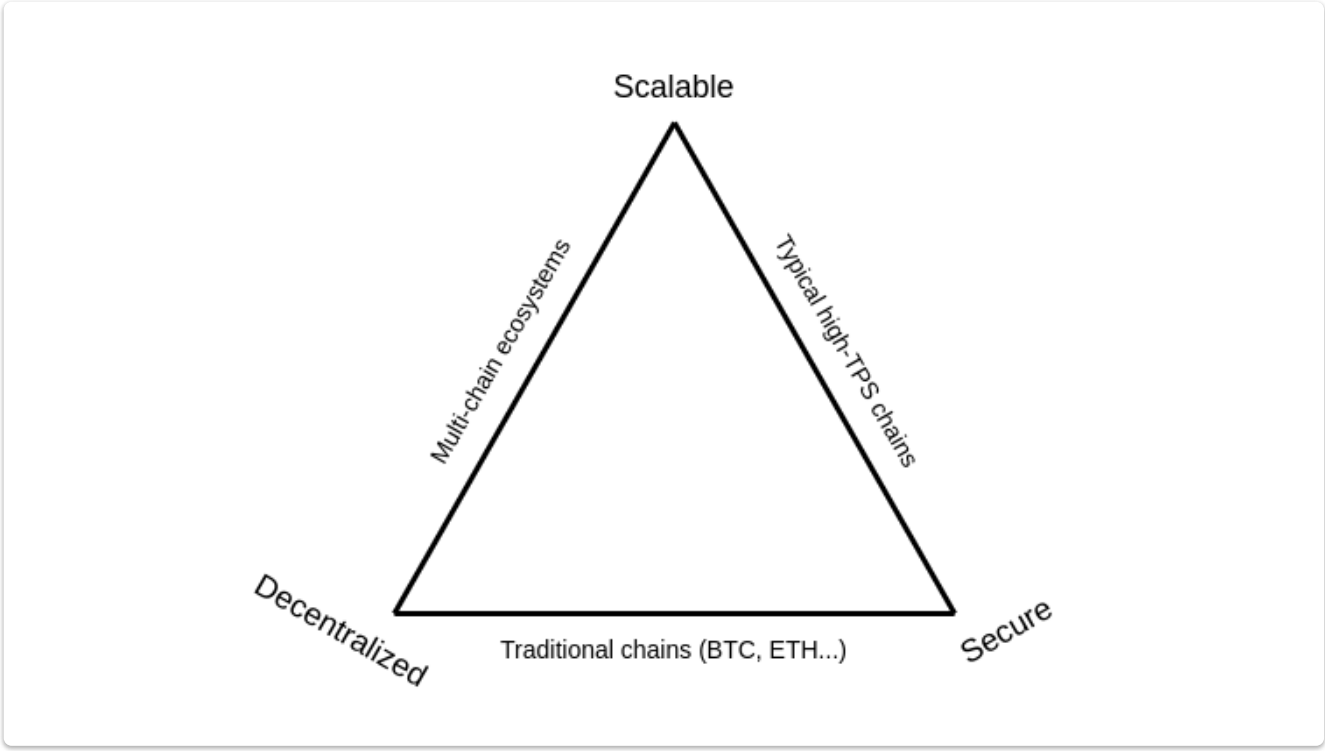
Privacy Preserving Financial Systems

Aztec



Their work has prompted the Ethereum Standard for Confidential Tokens : https://github.com/ethereum/EIPs/issues/1724

Blockchain Scalability

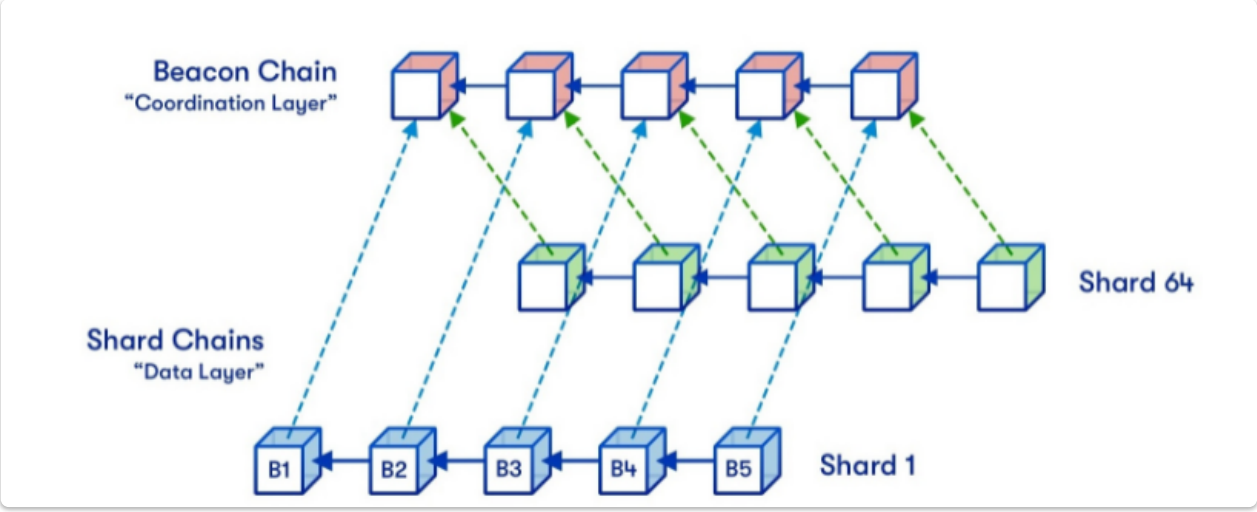


Approaches to increase scalability

ON CHAIN (L1) SOLUTIONS

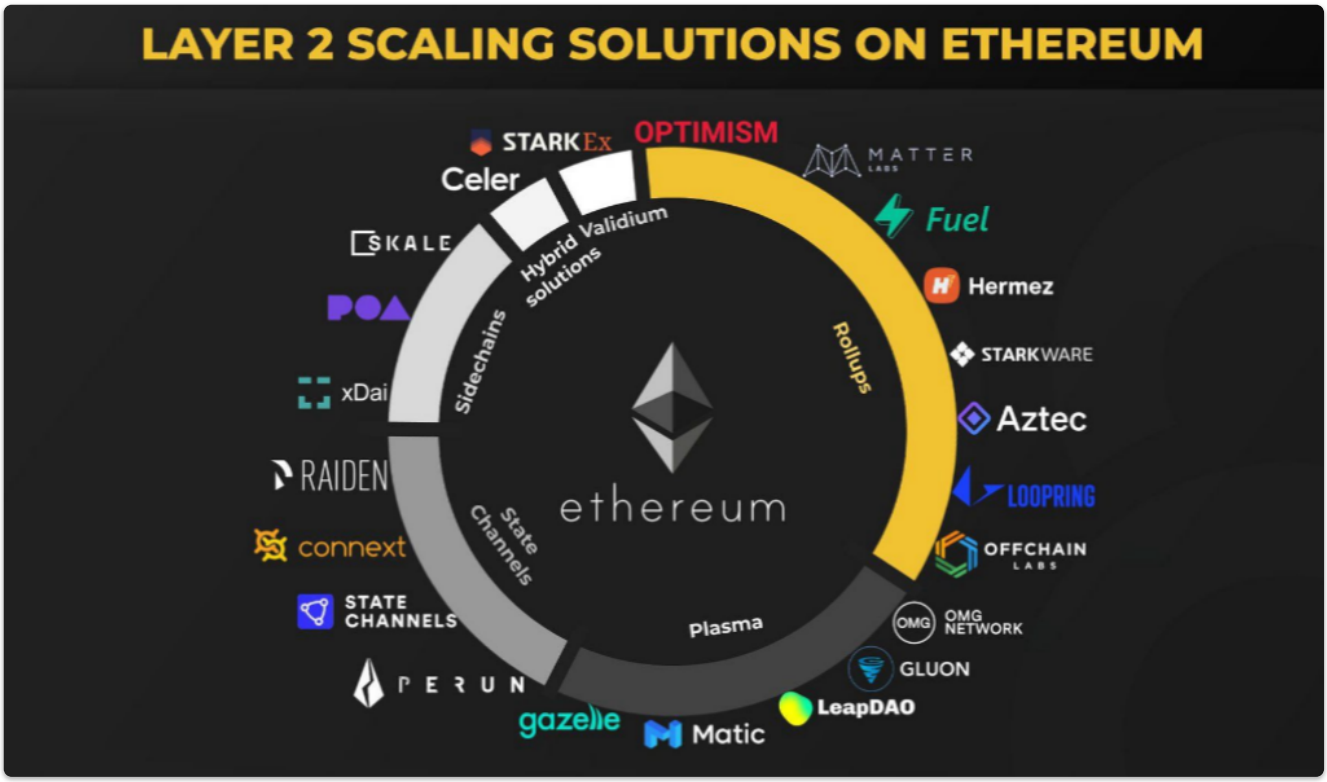
- Consensus mechanism
  Using DPoS - EOS
  PoH - Solana
  Snow etc. - Avalanche
  For example moving from Proof of Work to Proof of Stake - Ethereum
  The Beacon chain is already live, in 2022 there will be the merge of main net and the beacon chain.

- Sharding



OFF CHAIN SCALING (LAYER 2)

In essence transactions are submitted to these layer 2 nodes instead of being submitted directly to layer 1 (Mainnet).
For some solutions the layer 2 instance then batches them into groups before anchoring them to layer 1, after which they are secured by layer 1 and cannot be altered.
A specific layer 2 instance may be open and shared by many applications, or may be deployed by one project and dedicated to supporting only their application.



	State channels	Sidechains[0]	Plasma	Optimistic rollups	Validium	zkRollup
Security						
Liveness assumption (e.g. watch-towers)	Yes	Bonded	Yes	Bonded	No	No
The mass exit assumption	No	No	Yes	No	No	No
Quorum of validators can freeze funds	No	Yes	No	No	Yes	No
Quorum of validators can confiscate funds	No	Yes	No	No	Yes [1]	No
Vulnerability to hot-wallet key exploits	High	High	Moderate	Moderate	High	Immune
Vulnerability to crypto-economic attacks	Moderate	High	Moderate	Moderate	Moderate	Immune
Cryptographic primitives	Standard	Standard	Standard	Standard	New	New
Performance / economics						
Max throughput on ETH 1.0	1..∞ TPS [2]	10k+ TPS	1k..9k TPS [2]	2k TPS [3]	20k+ TPS	2k TPS
Max throughput on ETH 2.0	1..∞ TPS [2]	10k+ TPS	1k..9k TPS [2]	20k+ TPS	20k+ TPS	20k+ TPS
Capital-efficient	No	Yes	Yes	Yes	Yes	Yes
Separate onchain tx to open new account	Yes	No	No	No	No	No [5]
Cost of tx	Very low	Low	Very low	Low	Low	Low
Usability						
Withdrawal time	1 confirm.	1 confirm.	1 week [4] ([7])	1 week [4] ([7])	1..10 min [7]	1..10 min [7]
Time to subjective finality	Instant	N/A (trusted)	1 confirm.	1 confirm.	1..10 min	1..10 min
Client-side verification of subjective finality	Yes	N/A (trusted)	No	No	Yes	Yes
Instant tx confirmations	Full	Bonded	Bonded	Bonded	Bonded	Bonded
Other aspects						
Smart contracts	Limited	Flexible	Limited	Flexible	Flexible	Flexible
EVM-bytecode portable	No	Yes	No	Yes	Yes	Yes
Native privacy options	Limited	No	No	No	Full	Full

[0] Some researchers do not consider them to be part of L2 space at all, see https://twitter.com/gakonst/status/1146793685545304064

[1] Depends on the implementation of the upgrade mechanism, but usually applies.

[2] Complex limitations apply.

[3] To keep compatibility with EVM throughput must be capped at 300 TPS

[4] This parameter is configurable, but most researchers consider 1 or 2 weeks to be secure.

[5] Depends on the implementation. Not needed in zkSync but required in Loopring.

[7] Can be accelerated with liquidity providers but will make the solution capital-inefficient.

Matter Labs

Updated 2021–02–18

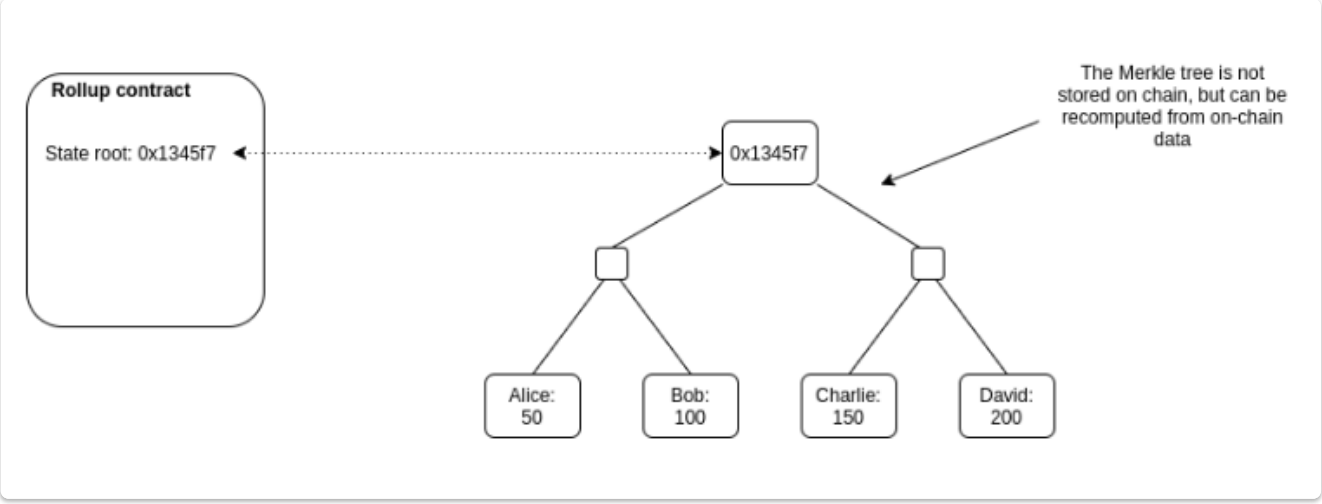
Rollups

Rollups are solutions that have

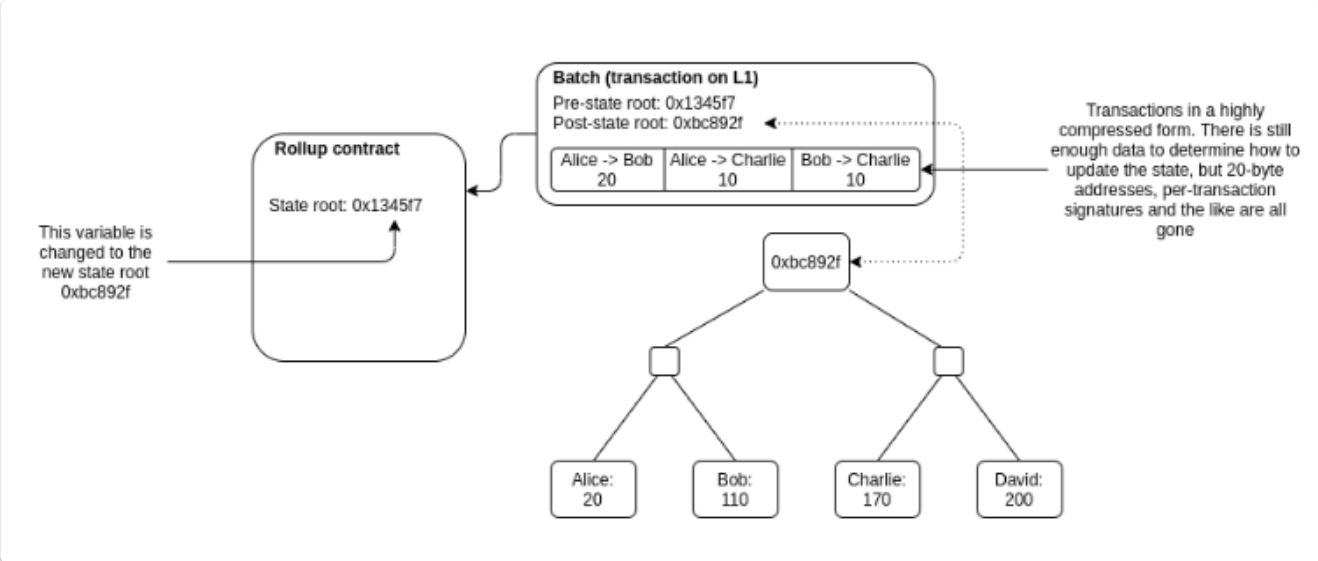
- transaction execution outside layer 1
- transaction data and proof of transactions is on layer 1
- a rollup smart contract in layer 1 that can enforce correct transaction execution on layer 2 by using the transaction data on layer 1

The main chain holds funds and commitments to the side chains
The side chain holds additional state and performs execution
There needs to be some proof, either a fraud proof (Optimistic) or a validity proof (zk)

Rollups require “operators” to stake a bond in the rollup contract. This incentivises operators to verify and execute transactions correctly.



Anyone can publish a batch, a collection of transactions in a highly compressed form together with the previous state root and the new state root (the Merkle root after processing the transactions). The contract checks that the previous state root in the batch matches its current state root; if it does, it switches the state root to the new state root.



There are currently 2 types of rollups

- Zero Knowledge Proof rollups
- Optimistic rollups

OPTIMISTIC ROLLUPS

The name Optimistic Rollups originates from how the solution works. 'Optimistic' is used because aggregators publish only the bare minimum information needed with no proofs, assuming the aggregators run without commiting frauds, and only providing proofs in case of fraud.
For more details see Arbitrum research

ZKP Rollups

An excellent report

An overview from Ethereum
The ZK-Rollup scheme consists of two types of users: transactors and relayers.

- Transactors create their transfer and broadcast the transfer to the network. The transfer data consists of an indexed “to” and “from” address, a value to transact, the network fee, and nonce. A shortened 3 byte indexed version of the addresses reduces processing resource needs. The value of the transaction being greater than or less than zero creates a deposit or withdrawal respectively. The smart contract records the data in two Merkle Trees; addresses in one Merkle Tree and transfer amounts in another.
- Relayers collect a large amount of transfers to create a rollup. It is the relayers job to generate the SNARK proof. The SNARK proof is a hash that represents the delta of the blockchain state. State refers to “state of being.” SNARK proof compares a snapshot of the blockchain before the transfers to a snapshot of the blockchain after the transfers (i.e. wallet values) and reports only the changes in a verifiable hash to the mainnet.

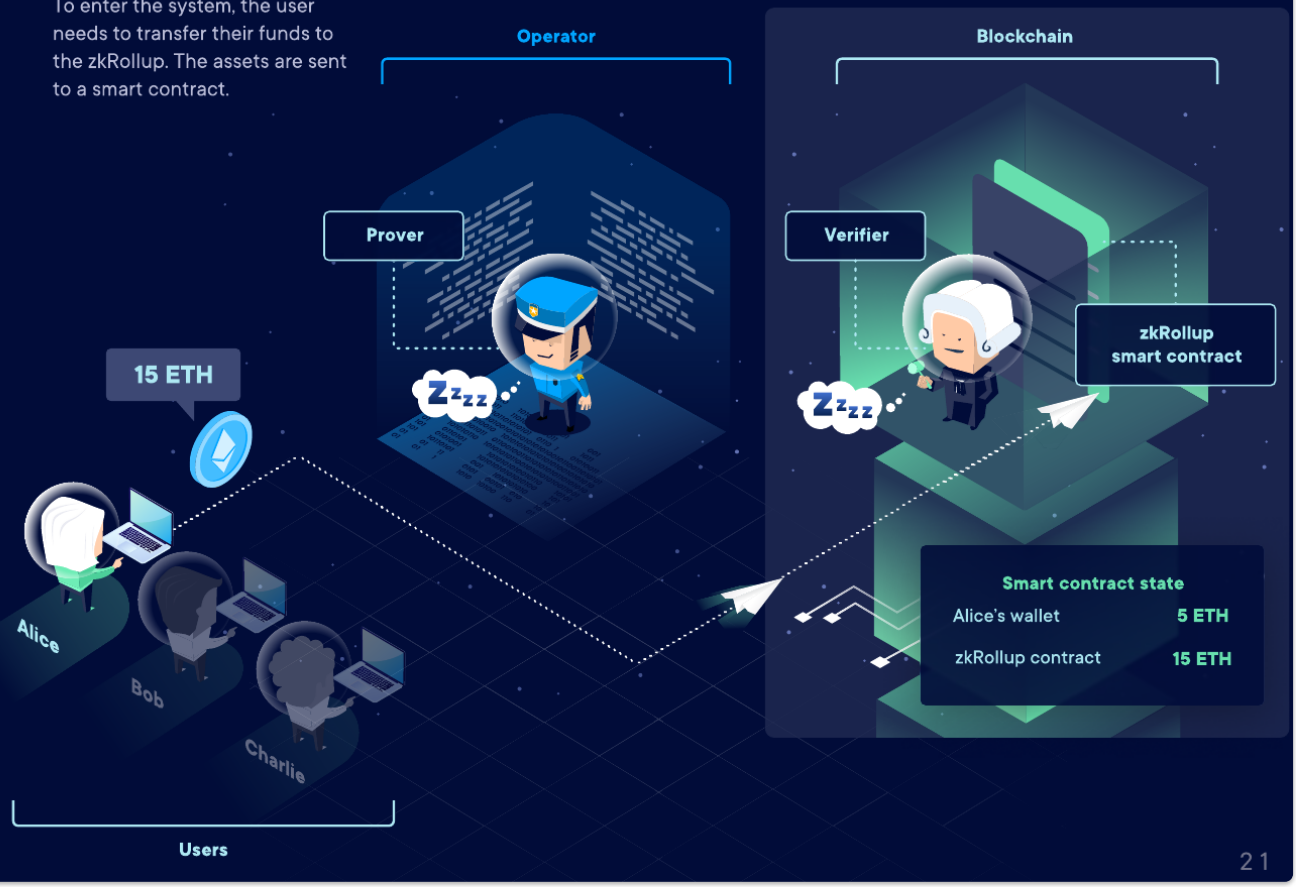
It is worth noting that anyone can become a relayer so long as they have staked the required bond in the smart contract. This incentivises the relayer not to tamper with or withhold a rollup.

ZK Rollup Process

From Ethworks

02. Alice's Enter

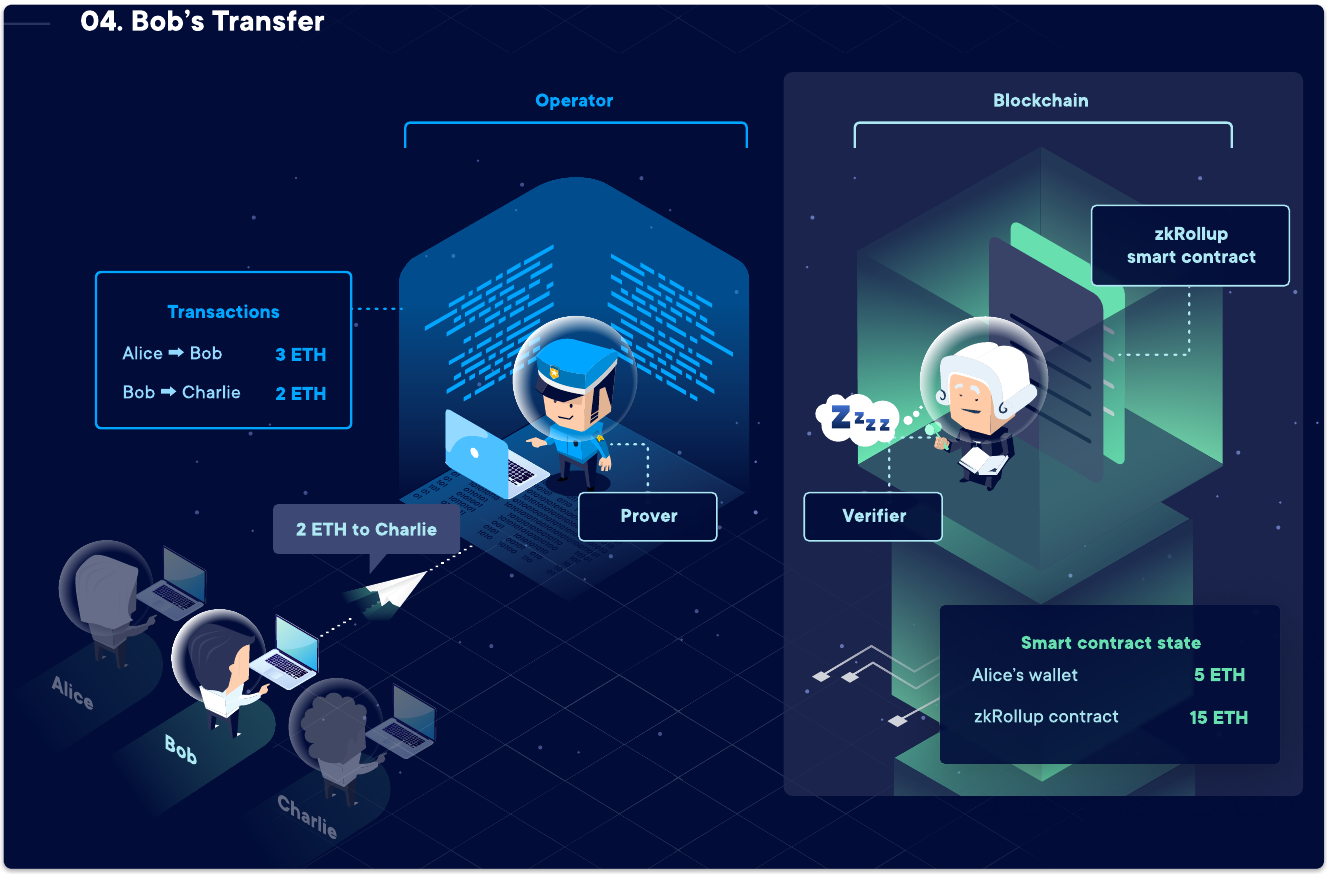
To enter the system, the user needs to transfer their funds to the zkRollup. The assets are sent to a smart contract.



03. Alice's Transfer

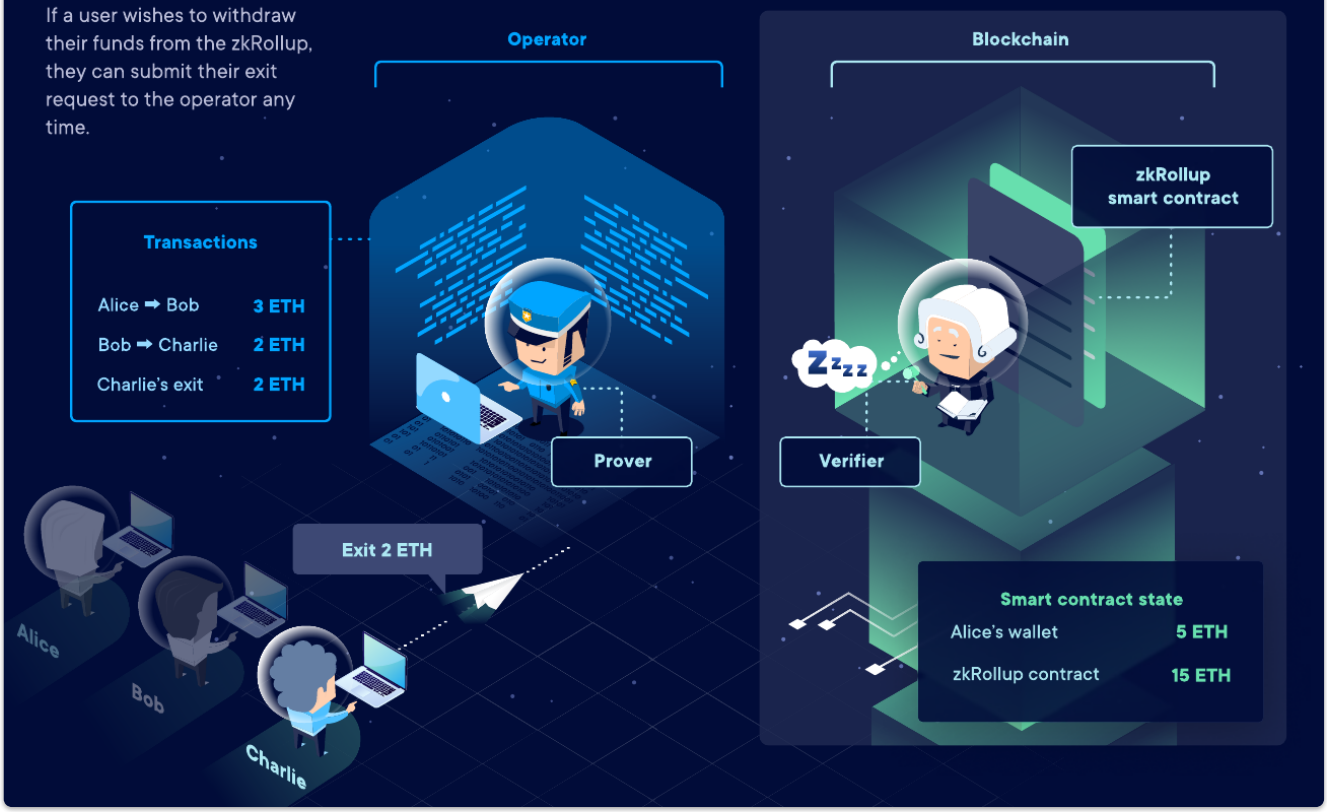
The user can now transfer their funds to another person. They sign the transaction and submit it to the zkRollup operator.

04. Bob's Transfer



05. Charlie's Exit

If a user wishes to withdraw their funds from the zkRollup, they can submit their exit request to the operator any time.



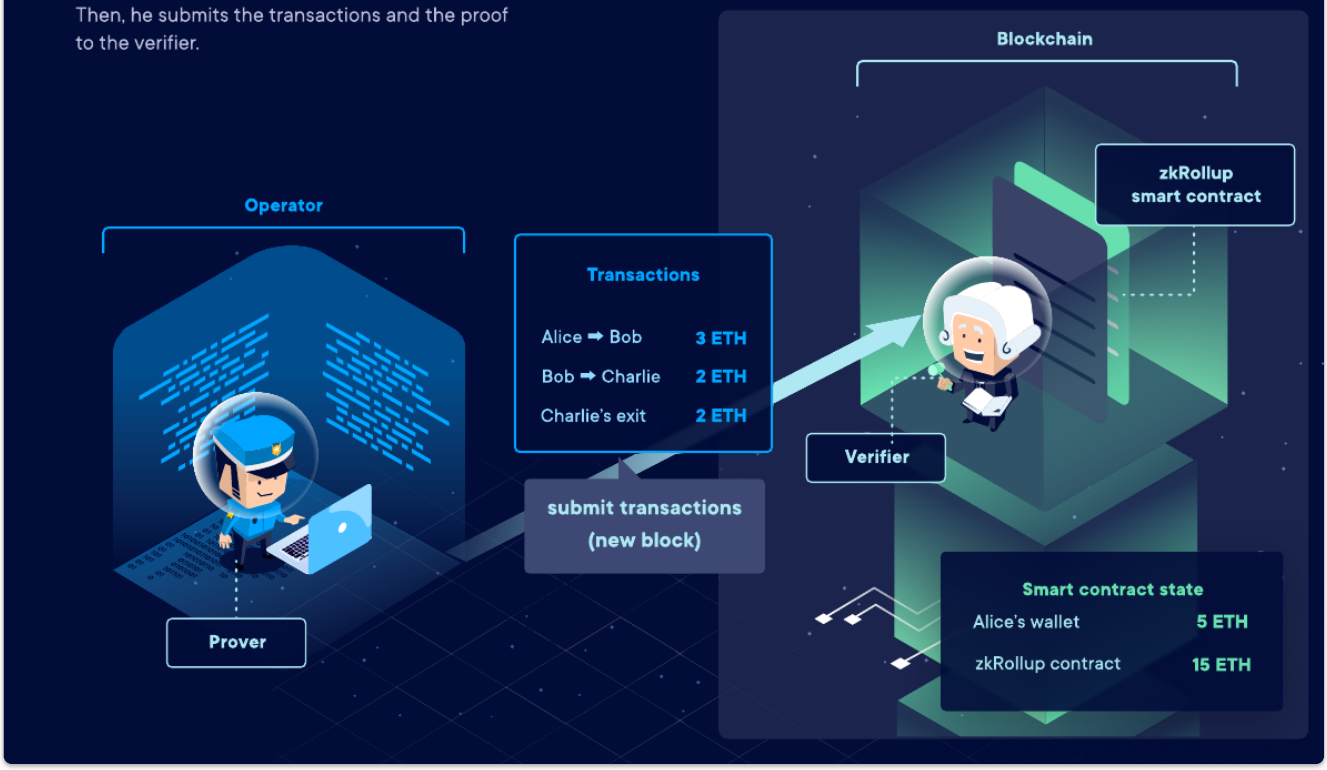
06. Collecting Transactions

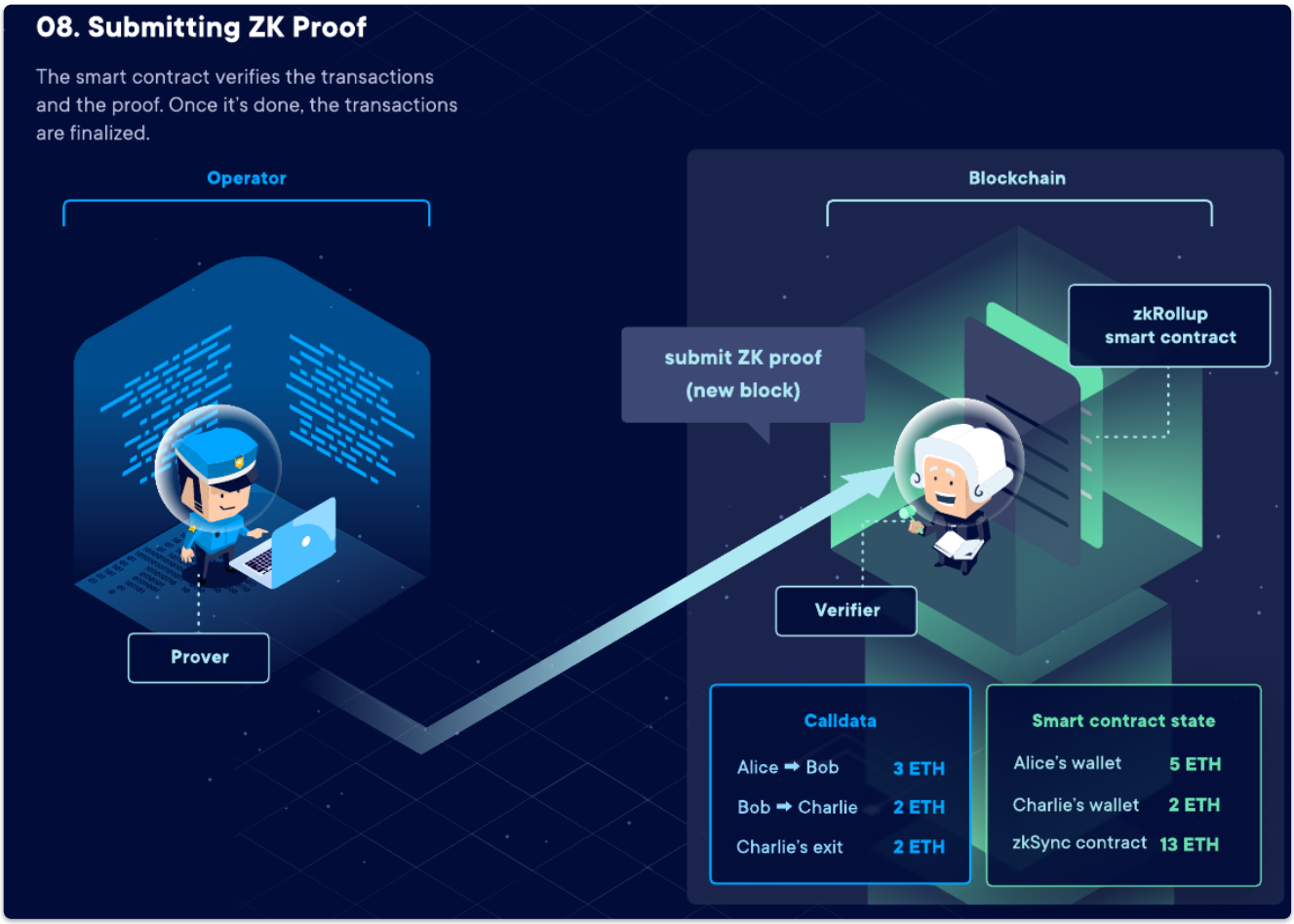
In the meantime, the operator collects transactions and exit requests from many users.

* Note that even if Bob and Charlie didn't have any funds on the zkRollup, they could still receive transfers from other users.

07. Submitting Transactions

Once in a while, the operator bundles the collected transactions together and generates a ZK proof. Then, he submits the transactions and the proof to the verifier.





Transaction Compression

How does compression work?

A simple Ethereum transaction (to send ETH) takes ~110 bytes. An ETH transfer on a rollup, however, takes only ~12 bytes:

Parameter	Ethereum	Rollup
Nonce	~3	0
Gasprice	~8	0-0.5
Gas	3	0-0.5
To	21	4
Value	~9	~3
Signature	~68 (2 + 33 + 33)	~0.5
From	0 (recovered from sig)	4
Total	~112	~12

Part of this is simply superior encoding: Ethereum's RLP wastes 1 byte per value on the length of each value. But there are also some very clever compression tricks that are going on:

Comparison of the types

Property	Optimistic rollups	ZK rollups
Fixed gas cost per batch	~40,000 (a lightweight transaction that mainly just changes the value of the state root)	~500,000 (verification of a ZK-SNARK is quite computationally intensive)
Withdrawal period	~1 week (withdrawals need to be delayed to give time for someone to publish a fraud proof and cancel the withdrawal if it is fraudulent)	Very fast (just wait for the next batch)
Complexity of technology	Low	High (ZK-SNARKs are very new and mathematically complex technology)
Generalizability	Easier (general-purpose EVM rollups are already close to mainnet)	Harder (ZK-SNARK proving general-purpose EVM execution is much harder than proving simple computations, though there are efforts (eg. Cairo) working to improve on this)
Per-transaction on-chain gas costs	Higher	Lower (if data in a transaction is only used to verify, and not to cause state changes, then this data can be left out, whereas in an optimistic rollup it would need to be published in case it needs to be checked in a fraud proof)
Off-chain computation costs	Lower (though there is more need for many full nodes to redo the computation)	Higher (ZK-SNARK proving especially for general-purpose computation can be expensive, potentially many thousands of times more expensive than running the computation directly)

Other approaches using zero knowledge

Recursive SNARKS - MINA

Mina is the first cryptocurrency protocol with a succinct blockchain.
With Mina, no matter how much the usage of the network grows, the blockchain always stays the same size - about 22kb.

Other projects

Filecoin

PROOF OF REPLICATION

In a Proof of Replication, a storage miner proves that they are storing a physically unique copy, or *replica*, of the data. Proof of Replication happens just once, at the time the data is first stored by the miner.

Whereas Proof of Replication is run once to prove that a miner stored a physically unique copy of the data at the time the sector was sealed, Proof of Spacetime (PoSt) is run repeatedly to prove that they are continuing to dedicate storage space to that same data over time.

Both the Proof of Replication and Proof of Spacetime processes in Filecoin use zk-SNARKs for compression.

The process of creating Filecoin's zk-SNARKs is computationally expensive (slow), but the resulting end product is small and the verification process is very fast. Compared to the original proofs, zk-SNARKs are tiny, making them efficient to store in a blockchain. For example, a proof that would have taken up hundreds of kilobytes on the Filecoin chain can be compressed to just 192 bytes using a zk-SNARK.

See zkSNARKS for the world

For storage to be verified on Filecoin, two proofs are involved: *Proof of Replication (PoRep)* and *Proof of Spacetime (PoSt).* In PoRep, a storage provider proves that they are storing a unique copy of a piece of data or information. PoRep happens just once, when the initial storage deal between client and provider happens and the data is first stored by the miner. Each PoRep that goes on-chain includes 10 individual SNARKs, which together prove that the process was done correctly through probabilistic challenges.

PoSt, on the other hand, serves to prove that the storage provider *continues* to store the original data over time without manipulation or corruption. When a storage provider first agrees to store data for a client, they must put down collateral in the form of FIL. If at any point during the agreement, the provider fails to prove PoSt, they are penalized and can lose all or some of their posted FIL collateral.

The result of an on-chain interaction in which the *prover* and *verifier* agree that data has been stored and maintained in an appropriate manner is a *proof.* As mentioned above, without a solution to make these proofs small and efficient, they would take up a tremendous amount of the network's bandwidth and deliver high operational costs to both storage providers and miners. By using zk-SNARKs to generate the proofs, however, the resulting proofs are small and the verification process is extremely fast (and thus, cheap). For example, proofs that typically would require hundreds of kilobytes to verify can instead

be compressed to just 192 bytes with zk-SNARKs. As mentioned above, each PoRep includes 10 SNARKs, meaning 1920 bytes in each (10 * 192 bytes).

Filecoin is the largest deployed zk-SNARK network to date

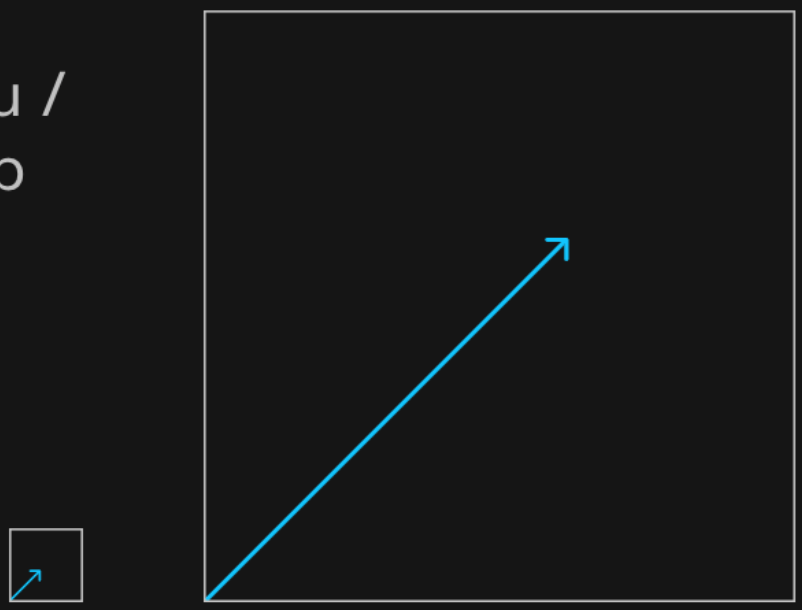
As far as we know, Filecoin represents the largest zk-SNARK deployment to date — in several dimensions:

- Our trusted setup enables circuits of up to 2^27 = ~134M constraints.
- Our large individual circuits have > 100M constraints.
- To satisfy our security requirements, some proofs bundle as many as 10 individual zk-SNARKs into a single large proof.
- We have also extended and deployed research on zk-SNARK aggregation to allow compression of thousands of individual proofs into a single proof.

All of the above contribute to Filecoin's ability to prove more information than the rest of the world has ever proved in production.

The baseline

Powers of Tau / Trusted Setup



Maximum Constraints

Zcash 2,097,152

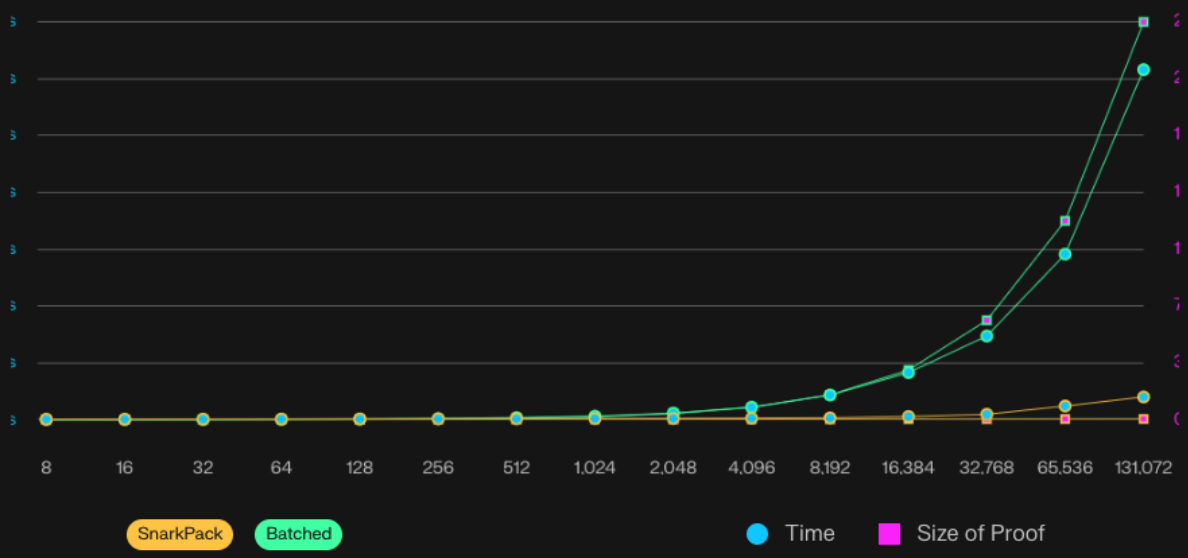
Filecoin 134,217,728

To support the amount of constraints needed for Filecoin, we ran a new Powers of Tau Ceremony, increasing the supported number by a factor of 64, over the Ceremony Zcash had run. This allows us to generate proofs of over 100 million constraints, limited only by the size of the parameters which must be distributed.

To support the Phase 2 (circuit-specific) trusted setup for our large circuits, we implemented techniques to dramatically decrease RAM usage, allow for parallelism, and reduce I/O overhead — in order to allow many parties using practical hardware to participate during the 7 weeks the ceremony took place.

Pack 'em tighter

SnarkPack



Even though Batch Verification helped, we needed faster verification, so we implemented SnarkPack. This allows us to aggregate many zk-SNARKs into a single combined proof. Not only does this optimization reduce verification time by a factor of more than 10x at scale — it also reduces chain bandwidth by reducing the average bytes-per-proof which must be submitted to the chain.

In order to accomplish this, we built on the research on the Inner Product Argument — and collaborated with the authors to extend it to support our needs without requiring a new trusted setup. We accomplished this by adapting the techniques to securely apply using two existing Powers of Tau trusted setups. This is a great example of how we have historically had to pick our way through obstacles to the practical realization of groundbreaking scale.

\

Tornado Cash

To process a deposit, Tornado.Cash generates a random area of bytes, computes it through the Pederson Hash (as it is friendlier with zk-SNARK), then send the token & the hash to the smart contract. The contract will then insert it into the Merkle tree.

To process a withdrawal, the same area of bytes is split into two separate parts: the **secret** on one side & the **nullifier** on the other side.
The nullifier is hashed.
This nullifier is a public input that is sent on-chain to get checked with the smart contract & the Merkle tree data. It avoids double spending for instance.

Thanks to a zk-SNARK, it is possible to prove the hash of the initial commitment and of the nullifier without revealing any information.
Even if the nullifier is public, privacy is sustained as there is no way to link the hashed nullifier to the initial commitment. Besides, even if the information that the transaction is present in the Merkle root, the information about the exact Merkle path, thus the location of the transaction, is still kept private.

Deposits are simple on a technological point of view, but expensive in terms of gas as they need to compute the hash & update the Merkle tree. At the opposite end, the withdrawal process is complex, but cheaper as gas is only needed for the nullifier hash and the zero-knowledge proof.

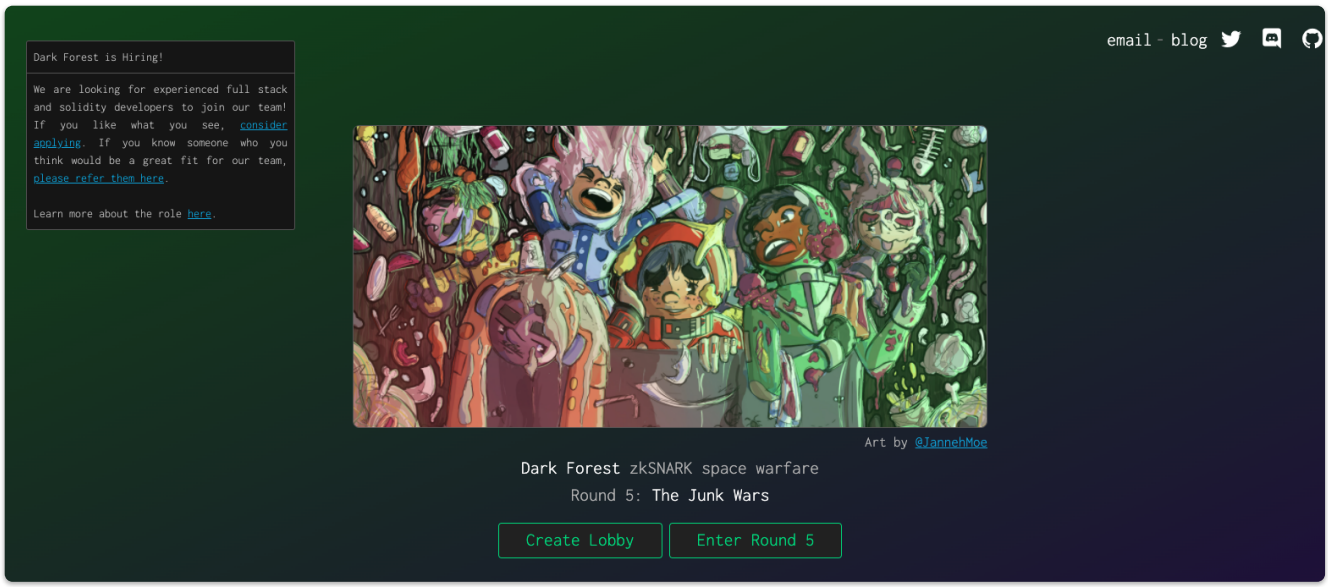
TORNADO CASH VERIFIER CONTRACT

Contract

Zero Knowledge Lottery based on Tornado Cash

See [article] (https://killari.medium.com/zero-knowledge-lottery-437e456dc3f2)

Dark Forest



How does Dark Forest use SNARKs?

A central mechanic in Dark Forest is that the cryptographic "fog of war." The fog of war ensures that you don't automatically know where all players, planets, and other points of interests are in the universe; you have to spend computational resources to discover them. This mechanic is secured by zkSNARKs.

In a universe with a fog of war, the locations of all players are private and hidden from each other. This means that players don't upload the coordinates of their planets to the Ethereum blockchain, which can be publicly inspected. Instead, each player uploads the hash of their location to the blockchain. This ensures that players stay "committed" to a specific location, but also that the location can't be determined from inspection of the Ethereum data layer.

Without zkSNARKs, there's an obvious attack vector - if a player uploads a random string of bytes that doesn't correspond to a real and valid location, and the integrity of the game is broken. To prevent this, Dark Forest requires players to submit zkSNARKs with every move to ensure that players are indeed submitting hashes corresponding to valid coordinates that they have knowledge of.

When players make moves, they're also required to submit ZK proofs that their moves are "valid" - you can't move too far or too fast. Without zkSNARKs, a malicious player could make illegal "teleport" moves by claiming that the hash they are moving from is next to the hash they're moving to, even if the two locations are actually on opposite sides of the universe. Once again, requiring ZK proofs keeps players honest. To use a chess analogy, the required ZK proofs basically tell the contract, "I'm moving my knight; I'm not going to tell you where I moved my knight from, or where I moved it to, but this proof proves that it did in fact move in a legal L-shape."

ID Escrow schemes

An Identity Escrow scheme allows users to identify themselves as members of a group with zero knowledge.

See Camenisch and Lysyanskaya for identity schemes using accumulators to prove set membership and non membership.

zkvSQL

See zkvSQL

a zero knowledge version of vSQL : verifiying SQL queries on outsourced databases

Other existing / suggested applications

- Verifying sufficient credit score - using range proofs
- Replacing username / passwords M-Pin
- Supply Chain Transparency Origin Trail
- Software Verification Paper
- Digital ForensicsPaper
- Identity Sovrin
- Verifying Qualification TiiQu
- KYC ING
- Tax Qedit
- Legal evidence integrity Stratuum