Mina Lecture

What is 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**.

What data is needed for usable representation of the blockchain ?

- balances of the accounts
- data that a node needs in order to verify that this state is real in a trustless manner
- the ability to broadcast transactions on the network to make a transfer

Recursive composition of proof (Succinctness)

> Blockchain is dynamic and new blocks keep getting added to it. However, we would like to ensure succinctness at any given point in time. Therefore, as the blockchain “grows”, we compute a new SNARK proof that not only validates the new blocks, but also the existing SNARK proof itself. The notion of a SNARK proof that attests to the verifiability of another SNARK proof is the notion of “incrementally-computable SNARK”

from *Mina: Decentralized Cryptocurrency at Scale (https://docs.minaprotocol.com/static/pdf/technicalWhitepaper.pdf)* (whitepaper)

Additional resourses about recursive SNARKs
https://zkproof.org/2020/06/08/recursive-snarks/

How Mina solves the verification problem

- SNARKS compress blocks into a single proof or certificate.
- End-users check this proof instead of checking the entire transaction history of a block.
- There is recursive composition of these SNARKs/certificates— enabling a constant sized blockchain.

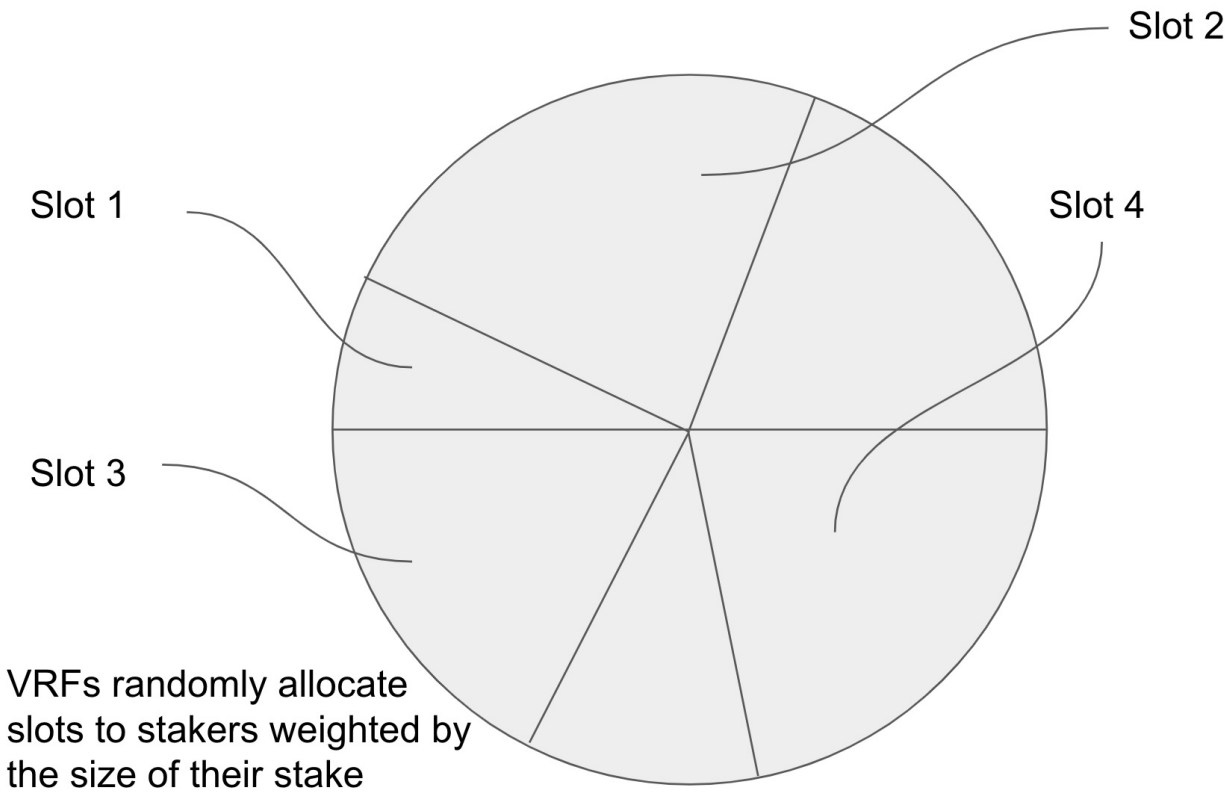
Mina's consensus mechanism

Mina's consensus mechanism is an implementation of Ouroboros Proof-of-Stake. Due to Mina's unique compressed blockchain, certain aspects of the algorithm have diverged from the Ouroboros papers, and the version Mina uses is called Ouroboros Samisik

VRFs are used to decide whether to produce a block, the probability is in proportion to the producer’s stake

The VRF in a produced block can be verified by the block recipients

All of the consensus rules are verified in a SNARK



Mina state

In addition to the SNARK ‘certificate’ verifying that the state transition from the previous block to the latest has been done correctly, the user can obtain a merkle proof of their current balance.

Generally we only need the latest state

A non-consensus node can store merkle paths to a number of account states

From this they are able to

- Check an account balance
- Create a transaction

The node can ask other nodes for any other account state

Block producers
Block producers are akin to miners or stakers in other protocols. By staking Mina, they can be selected to produce a block and earn block rewards in the form of coinbase, transaction fees and network fees. Block producers can decide to also be SNARK producers.

Snark workers
The second type of consensus node operator on Mina, snark producers help compress data in the network by generating SNARK proofs of transactions. They then sell those proofs to block producers in return for a portion of the block rewards

Professional block producers
Because staking requires nodes to be online, some may choose to delegate their Mina to staking pools. These groups run staking services in exchange for a fee, which is automatically deducted

when the delegator gets selected to be a block producer.

Is generating SNARKs similar to Proof-of-Work (PoW) mining?

No, they are different in several ways:

- SNARK work is deterministic, while PoW mining requires randomly calculating hashes to try and solve a puzzle. There is no luck element in SNARK work — if a snark worker wants to generate a SNARK of a transaction, they only need to generate the proof once. This means SNARK work is much less expensive and environmentally wasteful, as the compute is all spent towards a productive goal.
- There is no difficulty increase for SNARK work, as there is with PoW mining. In fact, as SNARK constructions, and proof generation times improve, the difficulty may actually decrease.
- SNARK work is not directly involved in consensus. Snark workers play no role in determining the next state of the blockchain. Their role is to simply generate SNARKs of transactions observed in the network
- As a snark worker, there is no requirement for uptime. PoW miners need to run their rigs non-stop to ensure they don't miss out on a potential block. Snark workers can come online and offline as they please — it is more like Uber, where there will always be work to be done, and nobody needs to say ahead of time when they want to work (polling vs callback).

zk-SNARK that Mina's zk-SNARK is called Pickles and recently it updated it's proof system to Kimichi.
Read more about those here:
Pickles
https://medium.com/minaprotocol/meet-pickles-snark-enabling-smart-contract-on-coda-protocol-7ede3b54c250
Kimichi
https://minaprotocol.com/blog/kimchi-the-latest-update-to-minas-proof-system

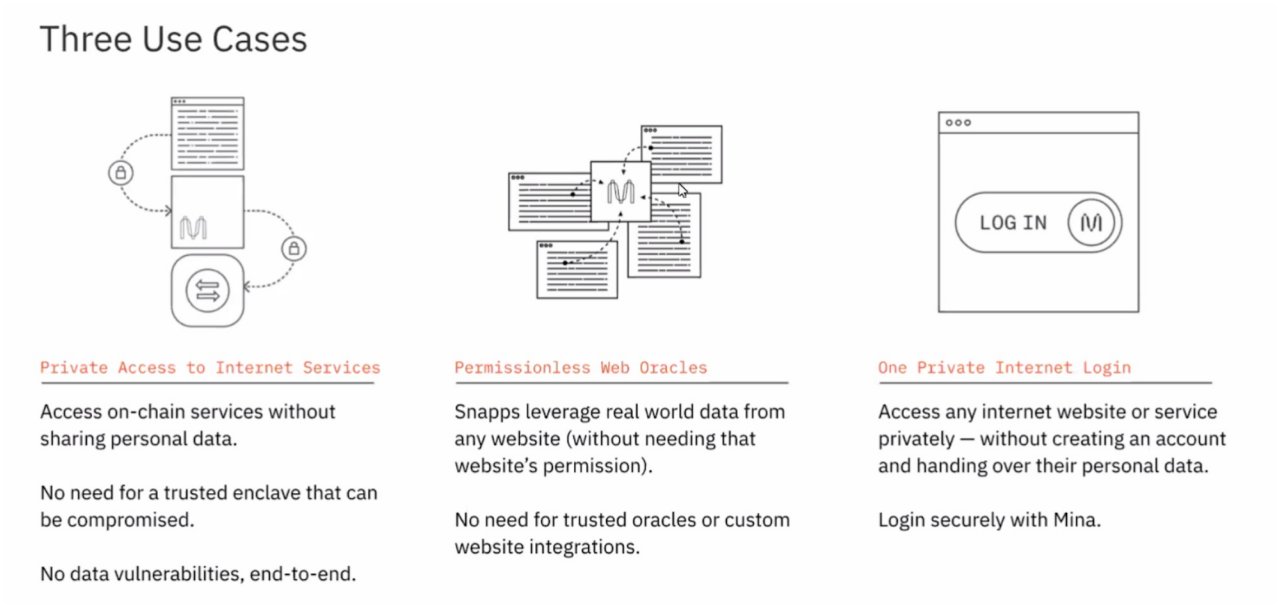
What are snapps?

> Snapps (“snark apps”) are Mina’s zk-SNARK-powered smart contracts.
> Snapps use an off-chain execution and mostly off-chain state model. This allows for private computation and state that can be either private or public.
> Using zk-SNARKs, snapps can perform arbitrarily complex computations off chain while incurring only a flat fee to send the resulting zero-knowledge proof to the chain, as opposed to other blockchains which use a gas-fee based model

Snapps = Dapps + Privacy + Off-Chain Data + Scalability
Snapps are featurewise similar to Dapps on Ethereum, but are superior thanks to three specific properties:

- Verify the integrity of a piece of data without disclosing what it is.
- Verify correct execution of expensive computations.
- Significant scalability benefits.

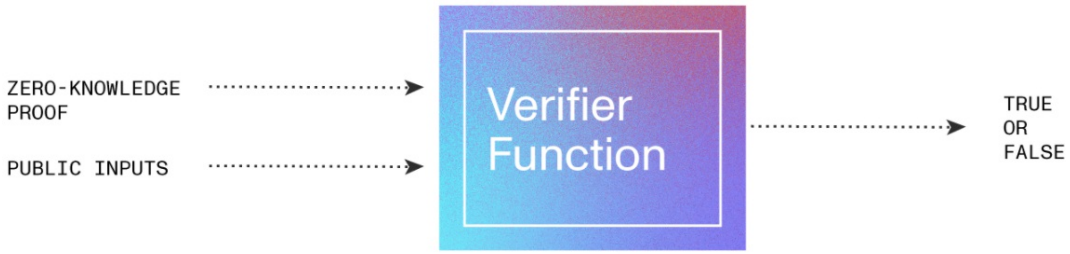
With a Snapp on Mina, the Snapp gets executed once by its developer, after which all other nodes can just verify the associated SNARK proof.



A snapp developer writes what a circuit, which is the method from which a prover function and a corresponding verifier function are derived during the build process.

The prover function runs in a user's web browser as part of the snapp. When interacting with a snapp's UI, users will enter any data required as input to the prover function, which will then generate a zero-knowledge proof.

The verification key lives on chain for a given snapp account and is used by the Mina network to verify that a zero-knowledge proof has met all constraints defined in the prover. A verification key is required in order to create a snapp account



To deploy a smart contract, means to put it on Mina network. To do this, the developer uses Snapp CLI. The deployment process sends a transaction containing the verification key to an address on the Mina blockchain.

When a Mina address contains a verification key, it acts as a snapp account. Whereas a regular Mina account can receive any transactions, a snapp account can only successfully receive transactions containing a proof that satisfies the verifier function. Any transactions that do not pass the verifier function will be rejected by the Mina network.

Docs: https://docs.minaprotocol.com/en/snapps

Steps to start development of Snapps
If running on your own machine the requirements are:
node.js and npm
https://github.com/nvm-sh/nvm

```
npm install -g snapp-cli
snapp project <name>
```

Browser IDE for snapps (pre-alpha -- doesn't seem to work currently)
https://editor.proxylabs.org/

Tick tac toe snapp
o1labs.org/tictactoe

Extropy is running Mina pool, if you're intresting in delgating your MINA tokens to us, take a look at the article we wrote !:
https://extropy-io.medium.com/staking-on-mina-fa540ad06811