
PLAYER-1
0
TOP SCORE
35000
PLAYER-2
0
TIME
2:13
zk-snarks!

zk-snark

- Zero Knowledge
- Succinct
- Non-interactive
- ARgument of Knowledge

“Zero-knowledge” proofs allow one party (the prover) to prove to another (the verifier) that a statement is true, without revealing any information beyond the validity of the statement itself. For example, given the hash of a random number, the prover could convince the verifier that there indeed exists a number with this hash value, without revealing what it is. (Zcash)

zk-snark

- “Programmable” zero knowledge proofs
- Compatible with ethereum via bn128
- Growing set of tools
- Compression / scalability
- Privacy & Anonymity
- Zcash / Filecoin / Coda / RollUp / ZEXE

“proof that I know how to do a computation”

selective information disclosure

Government issues to me a digital certificate (e.g. IDCAT)

- attributes: name, dni, birth date, ...
- public key associated to a private key under exclusive possession of owner
- digital signature of above made by government certificate

Generate a proof that I am able to compute (*π*)

- a digital certificate “X” is signed by government certificate
- the digital certificate “X” contains the public key “P”
- “P” public key matches with a private key “K”
- the digital certificate “X” birth date is before “DATE”

verify(*π, DATE)*

token privacy

I have a token transaction

- starting_balances db
- transaction account1 --(amount)--> account2
- ending_balances tdb

Generate a proof that I am able to compute (*π*)

- account1 start_balance >= amount
- account1 ending_balance = account1 start_balance - amount
- account2 ending_balance = account2 start_balance + amount

verify(*π, hash starting_balances_db, hash ending_balances_db)*

voting system

I have

- digital certificate issued by government
- starting votes db ---(my vote)---> ending votes db
- starting census db ---(my identity)---> ending census db

Generate a proof that I am able to compute (π)

- the certificate is valid and I am in posession of the associated private key
- ending_votes db party(my_vote) = starting_votes db party(my_vote) + 1
- all other voting count of another parties remains equal
- my certificate is the starting census db, and not in the ending census db

verify(π)

blockhain scalability (roll_up)

I have a lot of token transaction

- starting_balances db
- [transaction account1 --(amount)--> account2]*
- ending_balances tdb

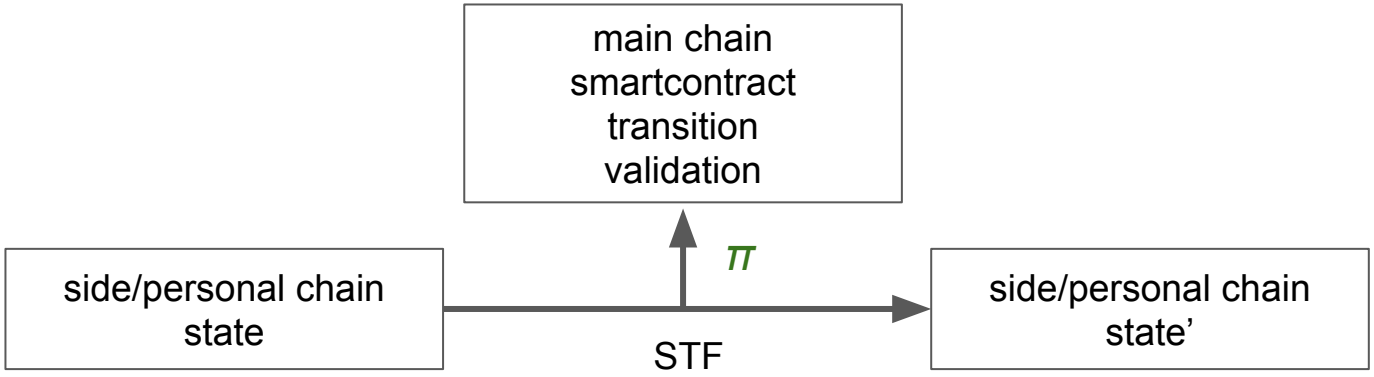
Generate a proof that I am able to compute (*π*)

- [account1 start_balance >= amount
- account1 ending_balance = account1 start_balance - amount
- account2 ending_balance = account2 start_balance + amount] *

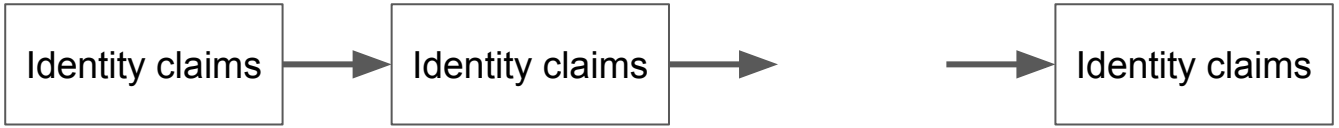
verify(*π, starting_balances_db, ending_balances_db, transactions)*

blockchain?

- Biggg problems: UX, Scalability, Privacy, Functional seg. & Legal Stuff (KYC)
- Going to be a “source of truth” for other structures
- Other structures: L2 Chains & Identity chains
- Assets (ether, tokens, etc…) and transaction validation
- Transaction validation: optimistic rollup & zk-rollup



identity chains



How transitions are verified?

- holochain : hash-links, self-signing, transitions validated by peers, DHT per app
- secure scuttlebut: hash-links, self-signing, LAN
- iden3: identity transition verifications batched in zk rollups
- caelum: agnostic

iden3 self-sovereign identity model

model

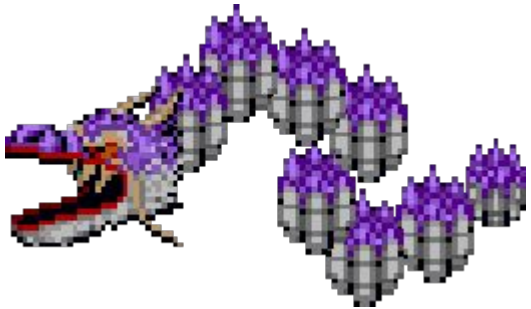
- custom tools
- hardcore-based in snarks circuits (babyjub, posseidon, …)
- identity chain of attributes issued to others
- identity chains integrity are secured by a roll_up in the main chain
- I can generate selective information disclosure proofs on my attributes

caelum self-sovereign identity model

model

- DECODE's ZenRoom LuaVM (cipher,sign,ecpairings)
- Focused on identity claims, more than identities itself
- Focus on proofs on generic DID Verifiable Claims
- Each proof has an associated validator that can be retrieved from provider
- Can generate selective information disclosure proofs on my attributes (coconout)

compilers & provers



- Zokrates
  - https://github.com/Zokrates/ZoKrates by Thibaut Schaeffer
  - Rust / Bellman
  - Algebra-approach, high level, can handle automatically a*b*c == 0
  - Focused on generating ethereum smartcontracts
- Za
  - https://github.com/adria0/za
  - Uses iden3's circomlang & circomlibs, circuit-approach, low level, allows manual optimizations
- Websnarks : `https://github.com/iden3/websnark`
- A lot more there...

twitter adria0

thanks!