Crypto-graphy and crypto-currencies

Séminaire cryptographie
Rennes
13 juin 2023

I have a small negative bias against blockckhains and cryptocurrencies otherwise somewhat neutral

Crypto-graphy and crypto-currencies

Séminaire cryptographie
Rennes
13 juin 2023

I have a small negative bias against blockckhains and cryptocurrencies
otherwise somewhat neutral
except frauds, scams, insider trades, price manipulation, pump-and-dump, pyramid schemes, Ponzi schemes, rug pull, pig butchering

Outline

Bitcoin 2018 paper

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non-reversible services. With the possibility of reversal, the need for trust spreads. Merchants must be wary of their customers, hassling them for more information than they would otherwise need. A certain percentage of fraud is accepted as unavoidable. These costs and payment uncertainties can be avoided in person by using physical currency, but no mechanism exists to make payments over a communications channel without a trusted party.

What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model[. . .]

What is needed is an electronic payment system based on cryptographic proof instead of trust, [. . .] without the need for a trusted third party

We propose a solution [. . .] using a peer-to-peer distributed timestamp server to generate [. . .] proof of the chronological order of transactions

Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Online, bitcoin.org/bitcoin.pdf. 2008

Meme

"It's the politics, stupid!"

JDVance, keynote speaker, Bitcoin Conference (May 2025, 35 000 participants)

Trump, keynote speaker, Bitcoin Conference (July 2024)

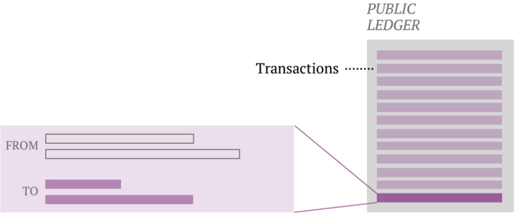
Transactions I

- Alice transfers bitcoins to Bob

From 1FGAftzSTztFSB8LN

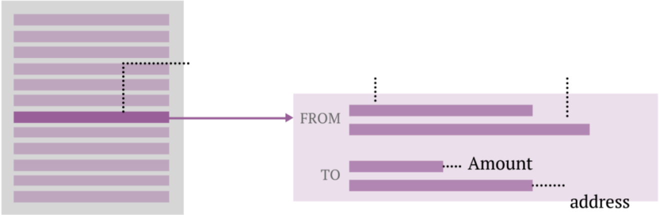
To 1CKNDhgvuX4T95T

- this is written in a public ledger



Transactions II

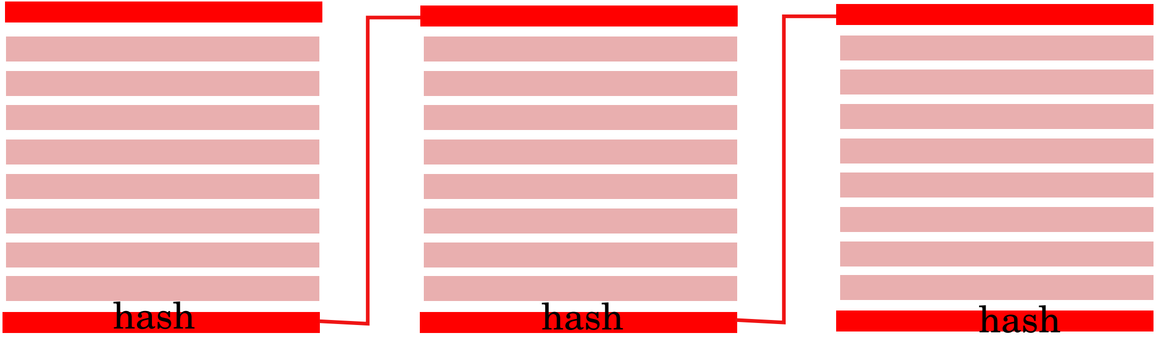
- Bob can then transfer to Carol



- simple
- combination of several inputs and outputs
- many, many outputs
- with coins to self

Bob has to sign the new transaction, with asymmetric cryptography
Bob’s ID is his public key

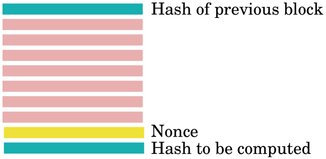
Blocks are chained by their hashes

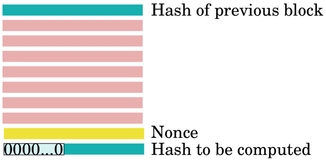


- the hash makes all past blocks (and transactions) immutable (collision resistance)
- 256 bits certifies 665 Gbytes of data
- “Proof-of-work Mining” is finding the hash
- Mining is done in a decentralized pseudonymous way

Proof-of-work Mining

Mining is finding a nounce wich contributes to a partially prescribed hash





nounce = an arbitrary number used only once

Proof-of-work

Proof of work (simplified)

given an integer N
to mine block-data = transactions:

```
nonce = 0
hash = H( block-data || nonce )
while hash does not start with N zero bits
    hash = H( block-data || nonce )
    nonce = nonce+1
```

- requiring the first bit to be zero: $\frac{1}{2}$ chance $\implies 2 = 2^1$ trials
- first two bits: $00, 01, 10, 11 \implies$ proba$= \frac{1}{4}$, $4 = 2^2$ trials
- number of trials for N bits: 2^N

Proof of work, more granularity

Define "target" $T \in [0, 2^{256})$

```
nonce = 0
hash = H( block-data || nonce )
while hash ≥ T    //hash is interpreted as an integer
    nonce = next nounce
    hash = H( block-data || nonce )
```

- Probability for success for one interation: $\frac{T}{2^{256}}$
- **Feedback**: T readjusted every 2016 blocks to keep block production rate at 10 min
- Current difficulty: Block 901051, June 13, 10:33:41

$$\texttt{0000000000000000000021e29ed399e23e1fa5a31929eae31acbd2cba6149e026} \sim 2^{78}$$

Cut-and-paste from Satochi Nakamoto’s paper

1. *new transactions are broadcast to all*† *nodes*
2. *each*† *node collects new transactions into a block*
3. *each*† *node works on finding a difficult proof-of-work for its block*
4. *when a*† *node finds a proof-of-work, it broadcasts the block to all nodes*
5. *nodes*† *accept the block only if all transactions in it are valid and not already spent*
6. *nodes*† **express their acceptance of the block** *by working on creating the next block in the chain, using the hash of the accepted block as the previous hash*

Longest chain rule

Nodes always consider the longest chain to be the correct one

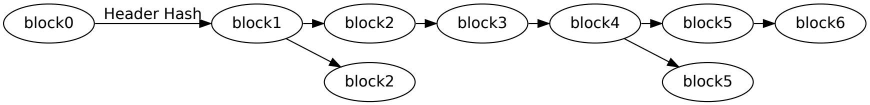
If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first.

In that case, they work on the first one they received, but save the other branch in case it becomes longer.

The tie is broken when the next proof-of-work is found and one branch is longer

Nodes that were working on the other branch will then switch to the longer one.

"Consensus" can change! $\Longrightarrow$ **"Eventual" consensus** (**"finality"** issues)



Orphaned blocks

Intro to smart contracts

- Register anything in the blockchain!
- Why not programs?
  - ...which then are immutable
  - ...which can be called by users' transactions
  - ...which are deterministically executed by block producers
  - ...which have their own persistent memory
  - ...which can hold, deliver and receive funds
- which can be installed ("deployed") by users
- which can be called by users...and by other programs
- **smart contracts**

Contract SchokoBueno

Generic transaction input data are fields: **msg.sender**
Contract explicit input (functions parameters): **destination** *and* **volume**

```
contract SchokoBueno {
    mapping(address => uint) SBBalance;

    function SchokoBueno(){// Endows creator  with 1000 units of SBs
        SBBalance[msg.sender] = 1000;
    }

    function send(address  destination, uint256  volume) {
        if ( volume <= SBBalance[msg.sender] ) {
          SBBalance[ msg.sender ] = SBBalance[ msg.sender ] -  volume;
          SBBalance[ destination] = SBBalance[ destination] +  volume;
        }
    }
}
```

Contract SchokoBueno with funds I

```
contract SchokoBueno {
    mapping ({address} => uint) SBBalance;
    address owner;

    function SchokoBueno () { // Set owner
        owner = msg.sender;
    }

    function buy() public payable {// msg.value to contract's balance
        SBBalance[ msg.sender ] = SBBalance[ msg.sender ] + msg.value / 100 ;
    }

    function sell(uint volume) public {
        if ( volume <= SBBalance[msg.sender]) {
            SBBalance[ msg.sender ] = SBBalance[msg.sender] - volume;
            msg.sender.transfer(volume * 100 ); // Ether to msg.sender
        }
```

Contract with SchokoBueno with funds II

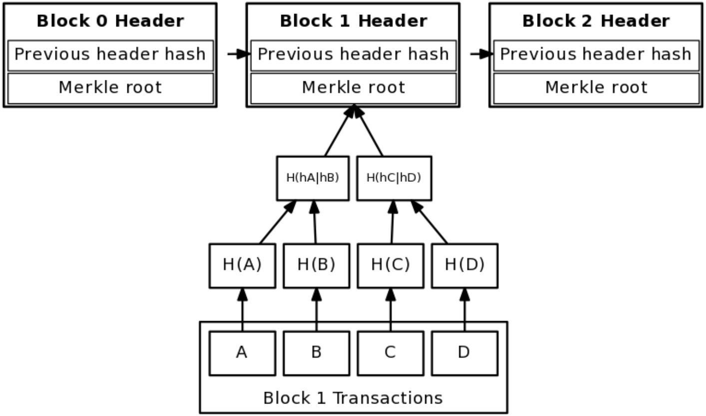
```
contract SchokoBueno {
    mapping (address => uint) SBBalance;
    address owner;

    function SchokoBueno () {
        owner = msg.sender;
    }

    function buy() public payable {// msg.value goes to contract's balance
        SBBalance[ msg.sender ] = SBBalance[ msg.sender ] + msg.value / 100 ;
    }

    function sell(uint volume) public {
        require(volume <= SBBalance[msg.sender])
        SBBalance[ msg.sender ] = SBBalance[ msg.sender ] - volume;
        msg.sender.transfer(volume * 100 ); // Ether to msg.sender
    }

    function takeTheMoneyAndRun() public {
        require(msg.sender == owner, "Only owner can steal the funds");
        msg.sender.transfer(address(this).balance);// Contract's Ether to owner
    }
}
```

DEFI “Decentralized finance” may not be that decentralized

```
function recall(address _from, uint256 _amount)
    external
    override
    onlyRegistrar
    returns (bool)
{
    require(
        _availableBalance(_from) >= _amount, // _amoun
        "SmartCoin: transfer amount exceeds balance"
    );
    super._transfer(_from, registrar, _amount);
    return true;
}
```

https://twitter.com/0xCygaar/status/1649108182957178882?s=20

Block headers



Merkle tree connecting block transactions to block header merkle root

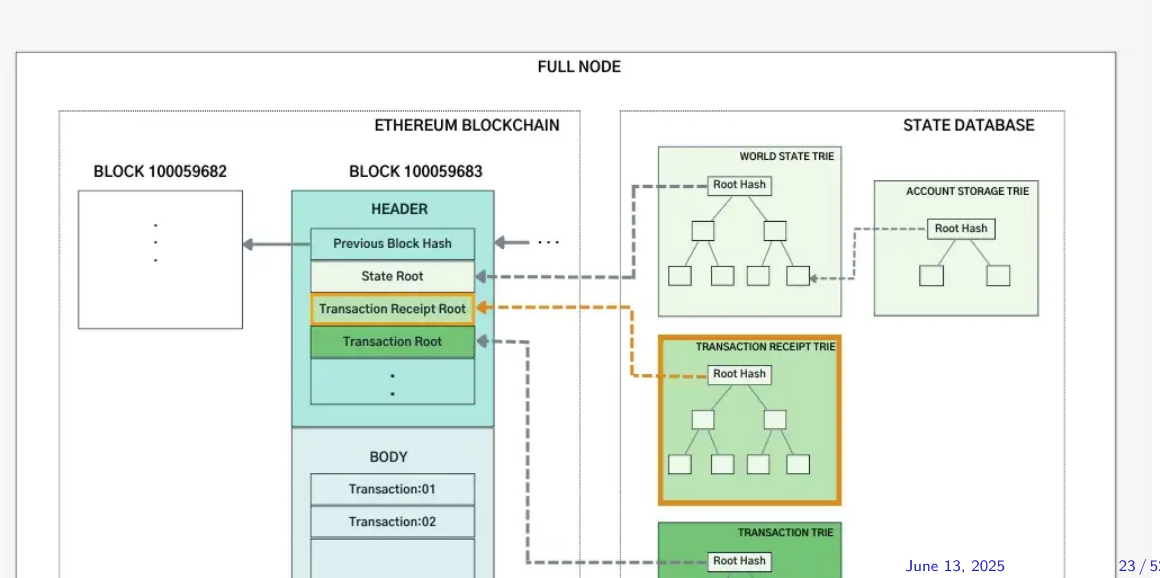
(Source: Bitcoin "white paper")

Simplified payment verification protocol (SPV)

Version	HashPrevBlock	HashMerkleRoot	TimeStamp	Target	Nonce
32 bits	256 bits	256 bits	32 bits	32 bits	32 bits

- A **lightweight client**
  - downloads the chain of block headers (thousand times smaller)
  - checks the proof of work of each block header
  - checks the chain of hashes
- For checking a transaction exists, a light client requests a **full node**
  - to provide the index of the block containing the transaction
  - to provide the Merkle branch (proof)

Ethereum headers I



Cryptographic ground of blockchains

1. Public key signature
2. Hash functions

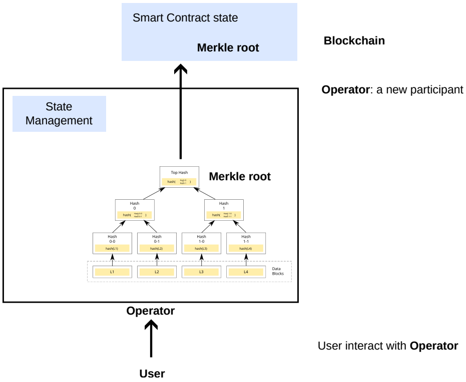
zk-SNARKS

- there is a **Prover**, there is a (weak) **Verifier**
- **Prover** wants to convince **Verifier that**
  - a statement t is true
  - **it** knows some secret s involved in the proof of t
  - without revealing s

Acronym zk-**S**-**N**-**AR**-**K**

- **Succinct**: A proof is very short and easy to verify.
- **Non-interactive**: No interaction, except the proof message
- **ARgument of Knowledge**: **Prover** is computationally bounded
- **zk**: zero-knowledge

Idea of a rollup



- The *operator* manages a list of user accounts
- Users send *offchain* transactions to *operator*
- The operator executes the transactions in batches
- The accounts (the *state*) are modified
- The operator store the root hash of the rollup state on the blockchain

Benefits

- One layer 1 transaction for many layer 2 transactions
- Compression for validity rollup:
  Only “functional data” sent to layer 1 (no signatures on Layer 1)
- Executing smart contracts **off-chain**
- No state management by validators

How to ensure the operator behaves well

Standard Ethereum blocks

- On Ethereum Layer 1, validator submit blocks with
  - a list of transactions
  - the new state root after execution of the transactions
- other validators redo the computations and accept if the new state root is correct

In a rollup, **the L2 operator** is **responsible of managing the state root**

How to ensure the operator behaves well

Standard Ethereum blocks

- On Ethereum Layer 1, validator submit blocks with
  - a list of transactions
  - the new state root after execution of the transactions
- other validators redo the computations and accept if the new state root is correct

In a rollup, **the L2 operator** is **responsible of managing the state root**

Optimistic Rollup

- In case of a fraud, a **fraud proof** is submitted and checked on Layer 1

How to ensure the operator behaves well

Standard Ethereum blocks

- On Ethereum Layer 1, validator submit blocks with
  - a list of transactions
  - the new state root after execution of the transactions
- other validators redo the computations and accept if the new state root is correct

In a rollup, **the L2 operator** is **responsible of managing the state root**

Optimistic Rollup

- In case of a fraud, a **fraud proof** is submitted and checked on Layer 1

ZK-rollup (validity rollup)

- The operator publishes on Layer1 a **validity proof** of correctness of new state
- Validity proofs can be checked by miners. There is an `ecPairing` Ethereum precompile

"ZK" stands for *zero-knowledge*, but is actually verifiable (no ZK) computation

Data availability: what is going on in the rollup

- **On chain**: some information about transactions is put on layer 1
  - Any one can reconstruct the state
  - “compressed form”: no need for transactions, much smaller adress space, etc.
- **Off chain**: no information about transactions is put on layer 1
  - No possibility to reconstruct the state

`https://l2beat.com`

`https://sorare.com`

Future of Ethereum

1. “The future of Ethereum will be rollup centric”
   $\implies$ modification of Ethereum L1 for better support
2. “blobs” for short-lived data availability

Verifiable computation for a given program F

Definition

- $(ek, vk) \leftarrow \textbf{KeyGen}(F)$
  - ek **evaluation key**
  - vk **verification key**
- $(\text{output } y, \text{proof } \pi) \leftarrow \textbf{Compute}(ek, \text{input } x)$
  - $y \leftarrow F(x)$
  - π proof that $y = F(x)$
- bool $b \leftarrow \textbf{Verify}(vk, x, y, \pi)$

Properties

- Completeness. If $y = F(x)$, $\textbf{Verify}(\mathtt{vk}, x, \pi) = \textbf{true}$
- Soundness. If $y \neq F(x)$, $\textbf{Verify}(\mathtt{vk}, x, \pi) = \textbf{false}$
- **Succintness**. $|\pi| = O(1)$
  (has constant size, depending only on the security parameter)

Arithmetic circuit for a program F

Field is $\mathbb{F}_r$

$F(x) = x^3 + x + 5$

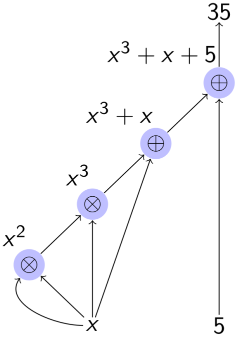


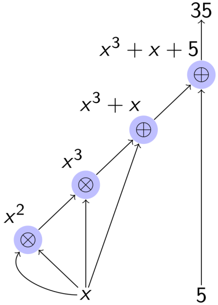
Image courtesy of Youssef El Housni

Theorem. Any program can be transformed into an arithmetic circuit

“I know x such that $y = F(x) = 35$” $\iff$ “I know wire values 3, 9, 27, 30, 35”

From circuit to matrices: “Rank-1 constraint system” (R1CS)

From $(3, 9, 27, 30)$ build $w = (1, 5, 3, 9, 27, 30, 35)$



$$(L_1 \cdot w^t)(R_1 \cdot w^t) = O_1 \cdot w^t$$
$$(L_2 \cdot w^t)(R_2 \cdot w^t) = O_2 \cdot w^t$$
$$(L_3 \cdot w^t)(R_3 \cdot w^t) = O_3 \cdot w^t$$
$$(L_4 \cdot w^t)(R_4 \cdot w^t) = O_4 \cdot w^t$$
$$(L_5 \cdot w^t)(R_5 \cdot w^t) = O_5 \cdot w^t$$

$$L = \begin{array}{c} \begin{matrix} 5 & 3 & 9 & 27 & 30 & 35 \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$R = \begin{array}{c} \begin{matrix} 5 & 3 & 9 & 27 & 30 & 35 \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$O = \begin{array}{c} \begin{matrix} 5 & 3 & 9 & 27 & 30 & 35 \end{matrix} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

From matrices to polynomials

$$L = \begin{array}{c} \begin{matrix} 5 & 3 & 9 & 27 & 30 & 35 \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} x \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{matrix} L_1(x) & L_2(x) & L_3(x) & L_4(x) & L_{27}(x) & L_6(x) & L_7(x) \end{matrix} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$R = \begin{array}{c} \begin{matrix} 5 & 3 & 9 & 27 & 30 & 35 \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} x \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{matrix} R_1(x) & R_2(x) & R_3(x) & R_4(x) & R_{27}(x) & R_6(x) & R_7(x) \end{matrix} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$O = \begin{array}{c} \begin{matrix} 5 & 3 & 9 & 27 & 30 & 35 \end{matrix} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{c} x \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{matrix} O_1(x) & O_2(x) & O_3(x) & O_4(x) & O_5(x) & O_6(x) & O_7(x) \end{matrix} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

$w = (1, 5, 3, 9, 27, 30, 35)$ is such that

$$\left(\sum_{i=1}^{6} w_i L_i(x_j) \right) \left(\sum_{i=1}^{6} w_i R_i(x_j) \right) = \sum_{i=1}^{6} w_i O_i(x_j), \quad j \in \{1, 2, 3, 4\}$$

A polynomial identity

$w = (1, 5, 3, 9, 27, 30, 35)$ such that

$$\left(\sum_{i=1}^{6} w_i L_i(x_j)\right)\left(\sum_{i=1}^{6} w_i R_i(x_j)\right) = \sum_{i=1}^{6} w_i O_i(x_j), \quad j \in \{1, 2, 3, 4\}$$

is such that

$$L(X)R(X) = O(X) \bmod Z(X)$$

with

$$L(X) = \sum_{i=1}^{6} w_i L_i(X), \quad R(X) = \sum_{i=1}^{6} w_i R_i(X), \quad O(X) = \sum_{i=1}^{6} w_i O_i(X)$$

$$Z(X) = \prod_{i=1}^{4} (X - x_i)$$

There exists $Q(X)$ such that

$$R(X)L(X) - O(X) = Q(X)Z(X)$$

Towards a short proof: Schwartz-Zippel

- Instead of requiring the verifier to check that

$$L(X)R(X) - O(X) = Q(X)Z(X)$$

- The equality is checked on a **single** random point $\tau \in \mathbb{F}$

$$L(\tau)R(\tau) - O(\tau) = Q(\tau)Z(\tau)$$

- Error probability is

$$\frac{d}{|\mathbb{F}_r|}$$

small with $|\mathbb{F}_r| = 2^{128}, 2^{256}$ and d is the number of gates

Hiding the evaluation point: computing "in the exponent"

$E(\mathbb{F}_p)$ which admits a subgroup $G_r = (\mathbb{Z}/r\mathbb{Z}, +)$
P a generator of G_r

- Verifier sends $L_i(\tau)P, R_i(\tau)P, O_i(\tau)P,\ \tau^i P,\ i = 1\ldots$
- **Without knowing** τ, Prover can compute and send

$$L(\tau)P,\ \ R(\tau)P,\ \ O(\tau)P,\ \ Q(\tau)P$$

  Example: $L(\tau)P = \left(\sum_i w_i L_i(\tau)\right) P = \sum_i w_i \cdot L_i(\tau)P$

- To check

$$L(\tau)R(\tau) = O(\tau) + Q(\tau)Z(\tau)$$

- Verifier checks, using a pairing $e(.,.)$

$$e(L(\tau)P, R(\tau)P) = e(O(\tau)P, P) \cdot e(Q(\tau)P, Z(\tau)P)$$

Zero-knowledge

The relation

$$L(X)R(X) = O(X) \bmod Z(X)$$

also holds for

$$(L(X) + \alpha Z(X))\,(R(X) + \beta Z(X)) = (O(X) + \gamma Z(X)) \bmod Z(X)$$

with α, β and γ random.

Checking the relation with

$$(L(\tau) + \alpha Z(\tau))\,(R(\tau) + \beta Z(\tau)) = (O(\tau) + \gamma Z(\tau)) = Q'(\tau)Z(\tau)$$

completely blinds $L(\tau)$, $R(\tau)$ and $O(\tau)$

Non interactive protocol with a "Trusted Setup" I

- circuit $\implies$ R1CS $\implies$ $L_i(X), R_i(X), O_i(X) \implies L_i(\tau), R_i(\tau), O_i(\tau)$, $i = 1 \dots$
- A **trusted authority** picks τ randomly in $\mathbb{F}$
- The "structured reference string" or "evaluation key" is

$$L_i(\tau)P, R_i(\tau)P, O_i(\tau)P, \tau^i P \quad i = 1, \dots$$

- Knowing the w_i's, the Prover can compute "in the exponent"

$$L(\tau)P,\ R(\tau)P,\ O(\tau)P,\ Q(\tau)P$$

  which makes the proof
- Verifier checks with a pairing $e(\cdot, \cdot)$

$$e(L(\tau)P, R(\tau)P) = e(O(\tau)P, P) \cdot e(Q(\tau)P, Z(\tau)P)$$

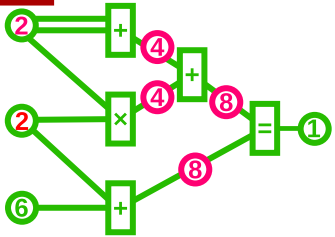
Technological pile

Program (frontend)

```
fn main() {
  let mv: Move = inputs.mv.parse();

  let pos = Chess::from(inputs.board);

  let mate = pos.play(&mv);

  assert! mate.is_checkmate();
}
```

1

Execution



2

Arithmetization

3

$$(\sum \gamma_i G(X))$$

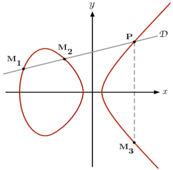
divides

$$(\sum \alpha_i A(X)) \times (\sum \beta_i B(X))$$

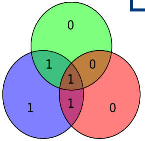
$$\textbf{mod } p \sim 2^{256} \sim 10^{77}$$

Crypto (backend)

4



Elliptic curves



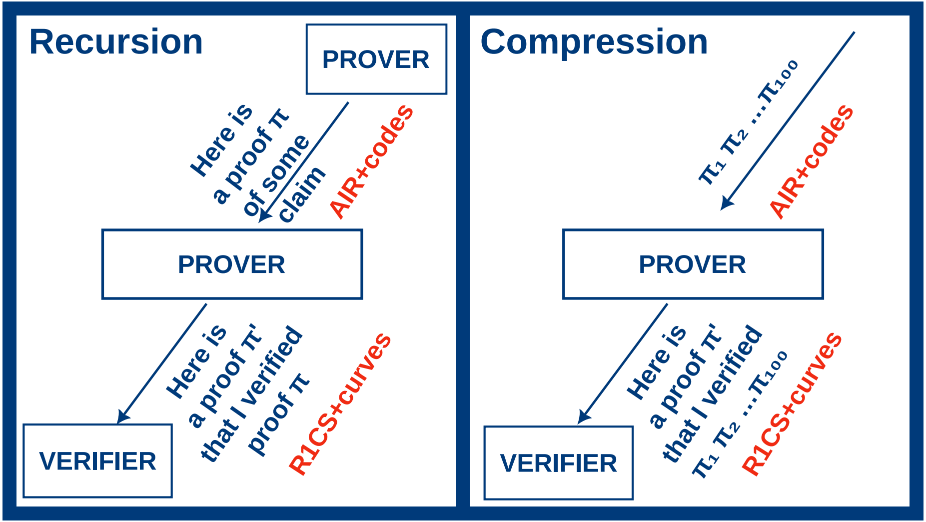
Error correcting codes

Combinatorial explosion (off-the-shelf software)

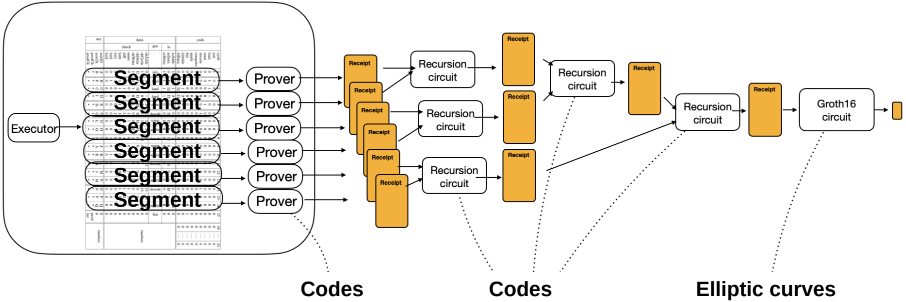
	Lang	Exec	Arith	Crypto
Gnark	Go	circuit	Plonk	curves
Winterfell	Rust	ciruit	AIR	codes
Arkworks	Rust	circuit	R1CS	curves
Plonky2	Rust	circuit	Plonk	codes
Halo2	Rust	circuit	Plonk	curves
Circom	Circom DSL	circuit	R1CS	curves
Bellman	Rust	circuit	R1CS	curves
Noir	Noir DSL	any	any	any
Cairo	Cairo DSL	Cairo VM	AIR	codes
Midem	Rust	Midem VM	AIR	codes
Matterlabs	Solidity	Ethereum VM	R1CS	curves
RISC0	Rust	RISCV	AIR	codes
Triton-VM		Triton-VM	AIR	codes

(disregarding curve choice, dedicated hardware, ASICs. . .)

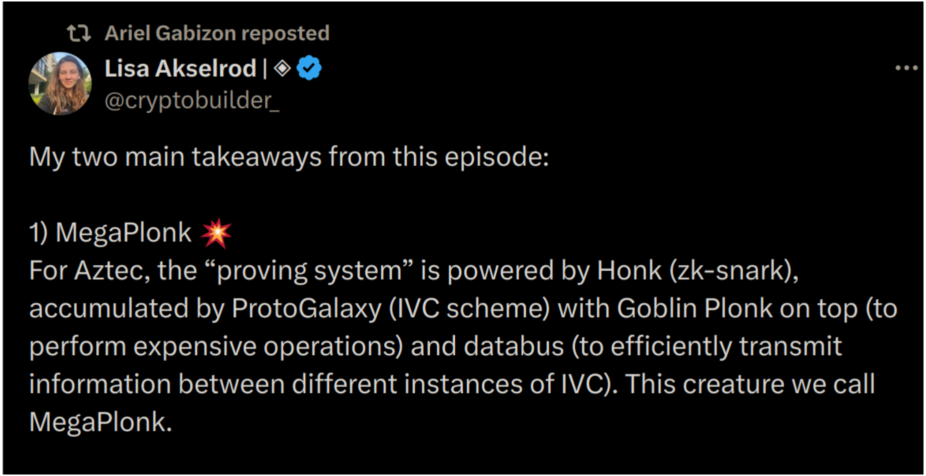
Proof of proof verification



RISC0


Executor
Segment
Segment
Segment
Segment
Segment
Segment
Prover
Prover
Prover
Prover
Prover
Prover
Receipt
Recursion circuit
Recursion circuit
Recursion circuit
Recursion circuit
Recursion circuit
Groth16 circuit
Codes
Codes
Elliptic curves

More combinatorial explosion


Ariel Gabizon reposted

Lisa Akselrod | ◈
@cryptobuilder_

My two main takeaways from this episode:

1) MegaPlonk 💥
For Aztec, the “proving system” is powered by Honk (zk-snark), accumulated by ProtoGalaxy (IVC scheme) with Goblin Plonk on top (to perform expensive operations) and databus (to efficiently transmit information between different instances of IVC). This creature we call MegaPlonk.

Recursion curves/curves: arithmetic mismatch

Statements over $\mathbb{F}_r$ are embedded into an additive group of order r of a curve $E(\mathbb{F}_p)$

Circuit is over F_r $\longrightarrow$ Verifier checks

$F(x) = x^3 + x + 5$

$$e(L(\tau)P, R(\tau)P) = e(O(\tau)P, P) \cdot e(Q(\tau)P, Z(\tau)P)$$

which is another circuit over $\mathbb{F}_p$

Verifier circuit is over F_p $\longrightarrow$ Verifier of verifier checks

$$e(L'(\tau)P', R'(\tau)P') = e(O'(\tau)P', P') \cdot e(Q'(\tau)P', Z'(\tau)P')$$

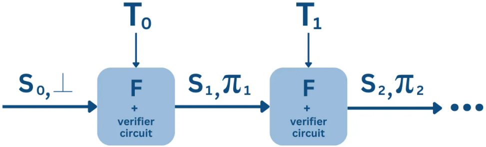
Infinite recursion requires to find (E, E') such that

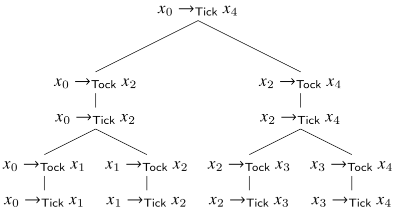
- $E(\mathbb{F}_p)$ admits a subgroup $(\mathbb{Z}/r\mathbb{Z}, +)$
- $E'(\mathbb{F}_r)$ admits a subgroup $(\mathbb{Z}/p\mathbb{Z}, +)$
- with pairings

Survey Aranha-Guillevic-ElHousni2022

Mina, by $O(1)$ labs

“22kB-Sized blockchain”





- “Tick snarks”, “tock snarks”
- Actually a cycle of pairing-friendly curves: MNT4 MNT6

“It’s the politics, stupid!”

Blockchains, “boring crypto”

Blockchains, “fancy crypto’

Conclusion

A lot of blockchain stuff $\perp$ to blockchains

1. "boring crypto" could be applied to any centralized system
   - committing to its state with public merkle root
   - chaining the hashes ($\approx$ git)
   - allowing users to get Merkle proof of their account
2. "fancy crypto" also could
   - making zk-proof of correcteness of hashes and transitions
   - enabling users to make zk-proof about their balance and transactions
3. recycle technology
   - Verifiable tls: zktls
   - Proof of software vulnerabilities: Cheesecoloth (Galois Inc)
   - zkml

Fed-up with cryptocurrencies?

- Cryptocurrencies are big: coinmarketcap.com
- Euro banknotes: €1,6 trillions
- No Internet? No problem!

Fed-up with cryptocurrencies?

- Cryptocurrencies are big: coinmarketcap.com
- Euro banknotes: €1,6 trillions
- No Internet? No problem!

"It's the politics, stupid!"

- Stablecoins
  - USDT (Tether). About $113 billion in holdings of US treasuries (Germany has $97 billion)

“It’s the politics, stupid!”

- Stablecoins
  - USDT (Tether). About \$113 billion in holdings of US treasuries (Germany has \$97 billion)
  - “We view [stablecoins] as a force multiplier of our economic might.” JD Vance, May 2025
  - USD1:
    - Trump stable coin
    - not be confused with TRUMP meme coin
    - managed by “World liberty financial” company

“Meet our team!”

“It’s the politics, stupid!”

- Stablecoins
  - USDT (Tether). About $113 billion in holdings of US treasuries (Germany has $97 billion)
  - “We view [stablecoins] as a force multiplier of our economic might” JD Vance, May 2025
  - USD1:
    - Trump stable coin
    - not be confused with TRUMP meme coin
    - managed by “World liberty financial” company
- Digital Yuan has smart contracts
- Digital euro, starting in October 2025