CS251 Fall 2025

(cs251.stanford.edu)

Privacy on the Blockchain

Dan Boneh

The need for privacy in the financial system

Supply chain privacy:

- A manufacturer does not want to reveal how much it pays its supplier for parts.

Payment privacy:

- A company that pays its employees in crypto wants to keep list of employees and salaries private.
- Endusers need privacy for rent, donations, purchases

Business logic privacy: Can the code of a smart contract be private?

The need for privacy in the financial system

The bottom line:

Blockchains cannot reach their full potential without some form of private transactions

Either (i) hide transaction amount, but not the end points, or

(ii) hide both the amount and the end points.

Types of Privacy

Pseudonymity: (weak privacy)

- Every user has a long-term consistent pseudonym (e.g. reddit)
  - Pros: reputation
  - Cons: link to real-world identity can leak over time

Full anonymity: User's transactions are unlinkable

- No one can tell if two transactions are from the same address

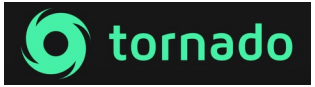
A difficult question: privacy from who?

No privacy:
Everyone can see all transactions in the clear

Privacy from the public:
Only a trusted operator can see transactions

Semi-full privacy:
only “local” law enforcement can see transactions

full privacy:
no one can see transactions



Negative aspects of full privacy

How to prevent criminal activity?

The challenge:

- How to support positive applications of private payments, but prevent the negative ones?
- Can we ensure legal compliance while preserving privacy?
- Yes! The key technology: **zero knowledge proofs**

Are Bitcoin and Ethereum Private?

The base systems are definitely not …

Privacy in Ethereum?

- Every account balance is public
- For Dapps: code and internal state are public
- All account transactions are linked to account

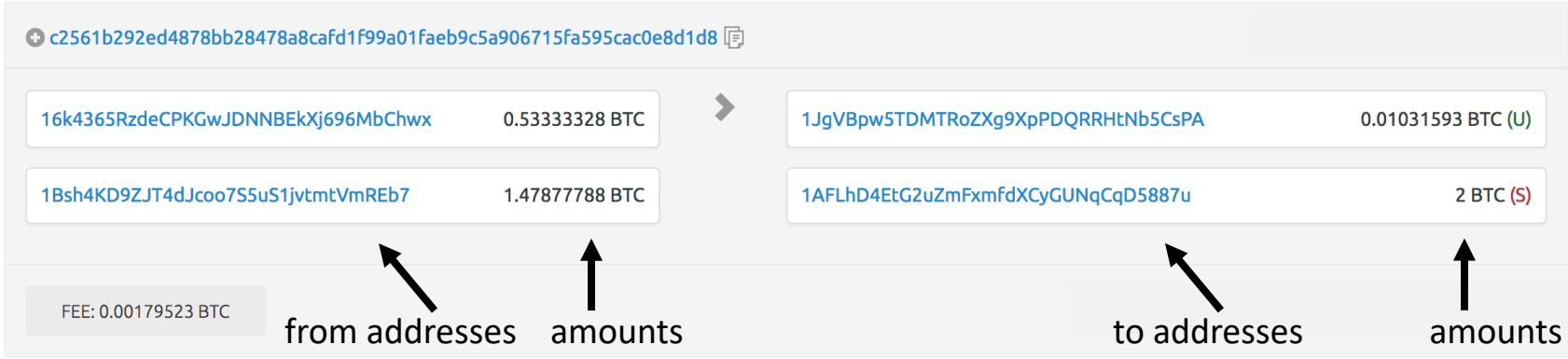
etherscan.io:

Address 0x1654b0c3f62902d7A86237…

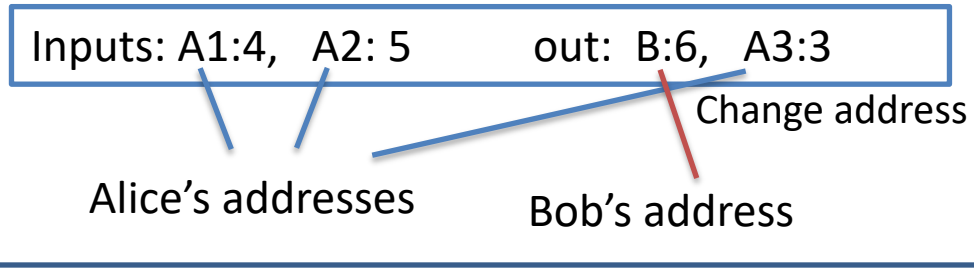
Balance:	1.114479450024297906 Ether
Ether Value:	$4,286.34 (@ $3,846.05/ETH)

	Txn Hash	Method ⓘ	Block
👁	0x0269eff8b4196558c07…	Set Approval For...	13426561
👁	0xa3dacb0e7c579a99cd…	Cancel Order_	13397993
👁	0x73785abcc7ccf030d6a…	Set Approval For...	13387834
👁	0x1463293c495069d61c…	Atomic Match_	13387703

Privacy in Bitcoin?



Alice can have many addresses (creating address is free)



Transaction data can be used to link an address to a physical identity

(chainalysis)

Linking an addresses to an identity

inputs: A1: 4, A2: 5 outputs: B: 6, A3: 3

Alice buys a book from a merchant:

- Alice learns one of merchant's address (B)
- Merchant links three addresses to Alice (A1, A2, A3)

Alice uses an exchange (ETH ↔ USD)

- BSA: a US exchange must do KYC (know your customer)
  … collect and verify Alice's ID
- Exchange links Alice to her addresses (A1, A2, A3)

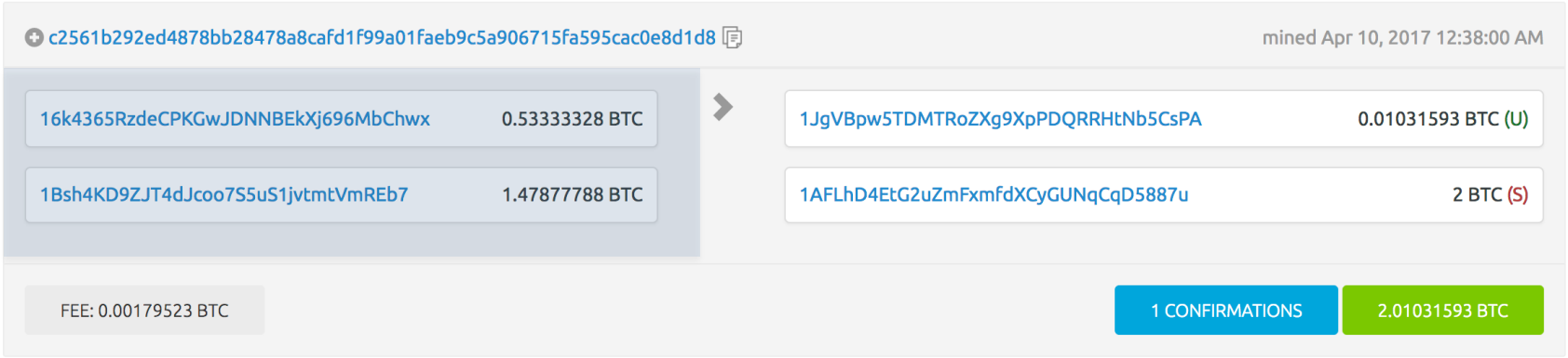
De-anonymization strategy: Idioms of use

A general strategy for de-anonymizing Bitcoin addresses

Heuristic 1:

Two addresses are input to a TX

⇒ both addresses are controlled by same entity



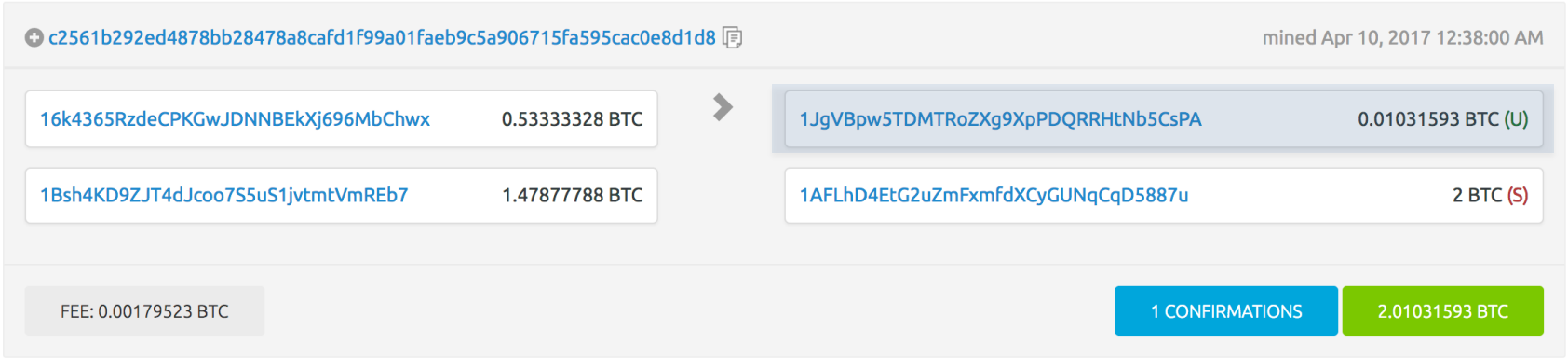
De-anonymization strategy: Idioms of use

Heuristic 2:

Change address is controlled by the same user as input address

Which is the change address?

- Heuristic: a new address that receives less than every input



A Bitcoin experiment [Meiklejohn, et al.]

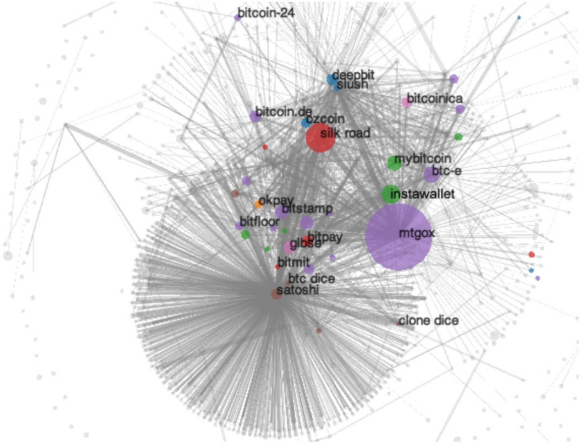
step 1: Heuristic 1 and 2 ⇒ 3.3M clusters

step 2: 1070 addresses identified by interacting with merchants

- Coinbase, Kraken, …

step 3: now 15% of all addresses identified

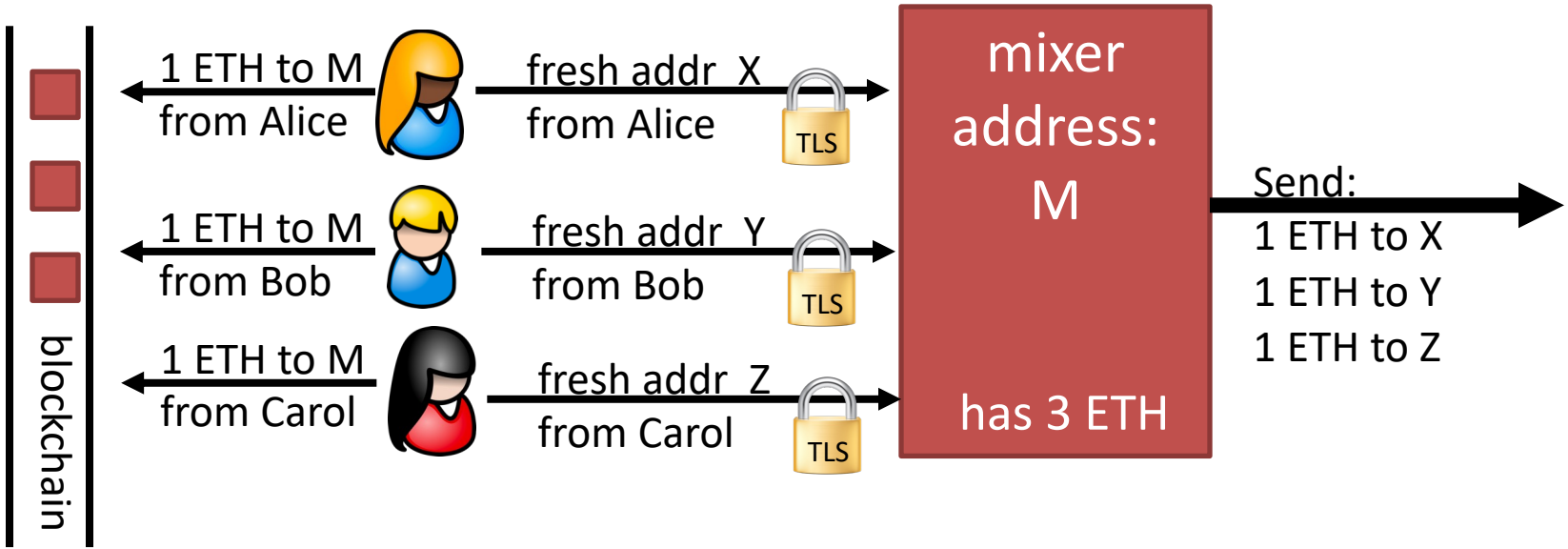
- Learn total assets for all clusters



Commercial efforts: Chainalysis, Elliptic, …

Private coins on a Public Blockchain

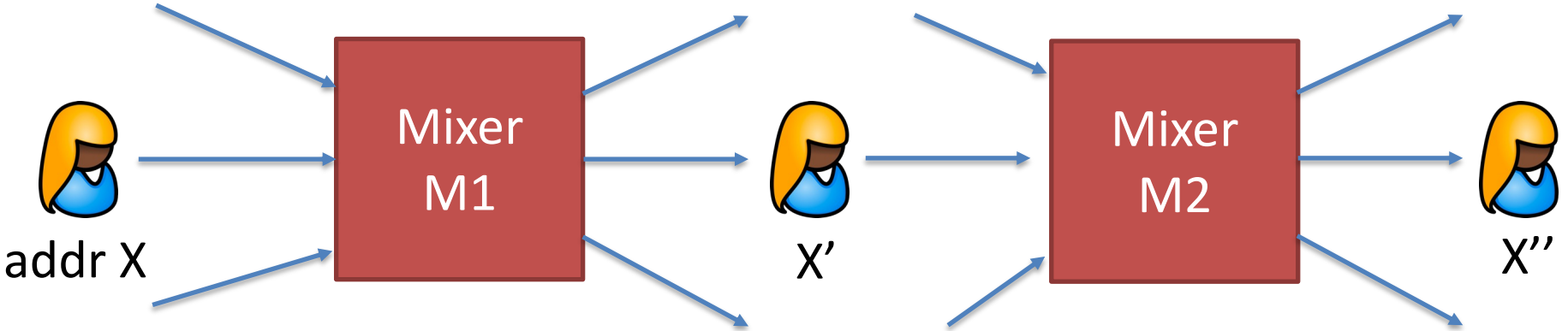
Attempt 1: simple mixing



Observer knows Y belongs to one of {Alice, Bob, Carol} but does not know which one
$\Longrightarrow$ anonymity set of size 3.

Problems: (i) mixer M knows shuffle, (ii) mixer can abscond with 3 ETH !!

Increasing the anonymity set



M1: mix n inputs from n users ⇒ X' has anonymity set size = n

M2: mix output from m mixers ⇒ X'' has anonymity set size = $n \times m$

Privacy: as long as one of M1 **or** M2 are honest

Secure mixing without a mixer?

Problem: Mixer can abscond with funds or reveal the shuffle.

Can we securely mix without a trusted mixer? Answer: yes!

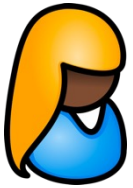
- on Bitcoin: **CoinJoin** (used by, e.g., Wasabi wallet)
- on Ethereum: **Tornado cash, Railgun, Privacy Pools, …**

… a single mixer using ZK proofs – next lecture

CoinJoin: Bitcoin Mixing without Mixer

The setup: Alice, Bob, and Carol want to mix together.

Alice owns UTXO **A1:5**, Bob owns UTXO **B1:3**, Carol owns **C1:2**



A1: 5, **A3** (change addr)

A2 (post mix address over Tor)

B1: 3, **B3** (change addr)

B2 (post mix address over Tor)

(same as Alice and Bob)

PASTEBIN

A1: 5, A3
B1: 3, B3
C1: 2, C3

B2, A2, C2

public forum

mix addresses

CoinJoin: Bitcoin Mixing without Mixer

CoinJoin TX: all three prepare and sign the following Tx:

inputs (not private):	A1: 5, B1: 3, C1: 2
outputs (private):	**B2: 2, A2: 2, C2: 2**
outputs (not private):	A3: 3, B3: 1

mix addresses

Mixed UTXOs all have same value = min of inputs (2 in this case)

All three post sigs on Pastebin ⇒ one of them posts Tx on chain.

Coinjoin drawbacks

In practice: each CoinJoin Tx mixes about 40 inputs

- Large Tx: 40 inputs, 80 outputs

All participants must sign CoinJoin Tx for it to be valid

⇒ ensures all of them approve the CoinJoin Tx

… but any one of them can disrupt the process

Beyond simple mixing

Private Tx on a public blockchain

Can we have private transactions on a public blockchain?

Naïve reasoning:

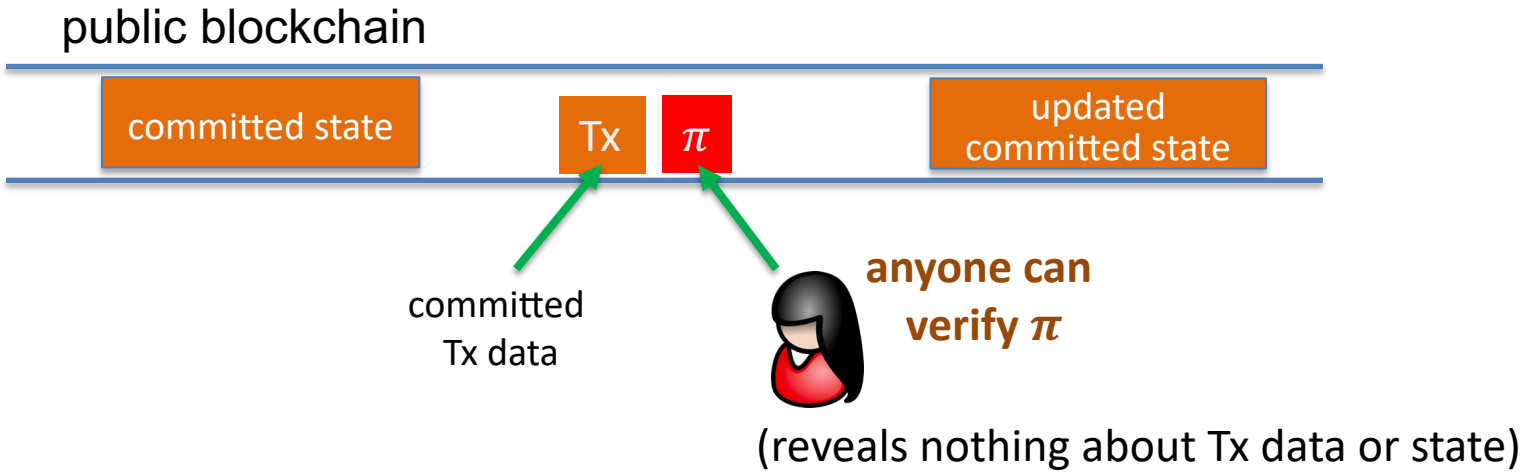
universal verifiability ⇒ transaction data must be public

otherwise, how we can verify Tx ??

crypto magic ⇒ private Tx on a publicly verifiable blockchain

Crypto tools: **commitments** and **zero knowledge proofs**

A paradigm for Private Tx



Committed data: short (hiding) commitment on chain

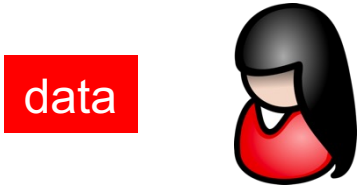
Proof π: succinct *zero-knowledge proof* that

(1) committed Tx data is consistent with committed current state, and

(2) committed updated state is correct

Review: cryptographic commitments

Cryptographic commitment: emulates an envelope



Many applications: e.g., a DAPP for a sealed bid auction

- Every participant **commits** to its bid,
- Once all bids are in, everyone opens their commitment

Cryptographic Commitments

Syntax: a commitment scheme is two algorithms

- **commit**(*msg*, *r*) → *com*

- **verify**(*msg*, *com*, *r*) → accept or reject

  anyone can verify that commitment was opened correctly

Commitments: security properties

- **binding**: Bob cannot produce two valid openings for ***com***

  More precisely: no efficient adversary can produce

  com, (m_1, r_1), (m_2, r_2)

  such that verify(m_1, ***com***, r_1) = verify(m_2, ***com***, r_2) = accept

  and $m_1 \neq m_2$.

- **hiding**: ***com*** reveals nothing about committed data

  commit(m, r) $\rightarrow$ ***com***, and r is sampled uniformly in a set R,

  then ***com*** is statistically independent of m

Example: hash-based commitment

Fix a hash function $H\colon M \times R \rightarrow C$ (e.g., SHA256)

where H is collision resistant, and $|R| \gg |C|$

- commit($m \in M$, $r \leftarrow R$): $\boldsymbol{com} = H(m, r)$
- verify(m, ***com***, r): accept if $\boldsymbol{com} = H(m, r)$

binding: follows from collision resistance of H

hiding: follows from a mild assumption on H

What is a zk-SNARK?

Succinct zero knowledge proofs:
an important tool for privacy on the blockchain

What is a zk-SNARK ? (intuition)

SNARK: a succinct proof that a certain statement is true

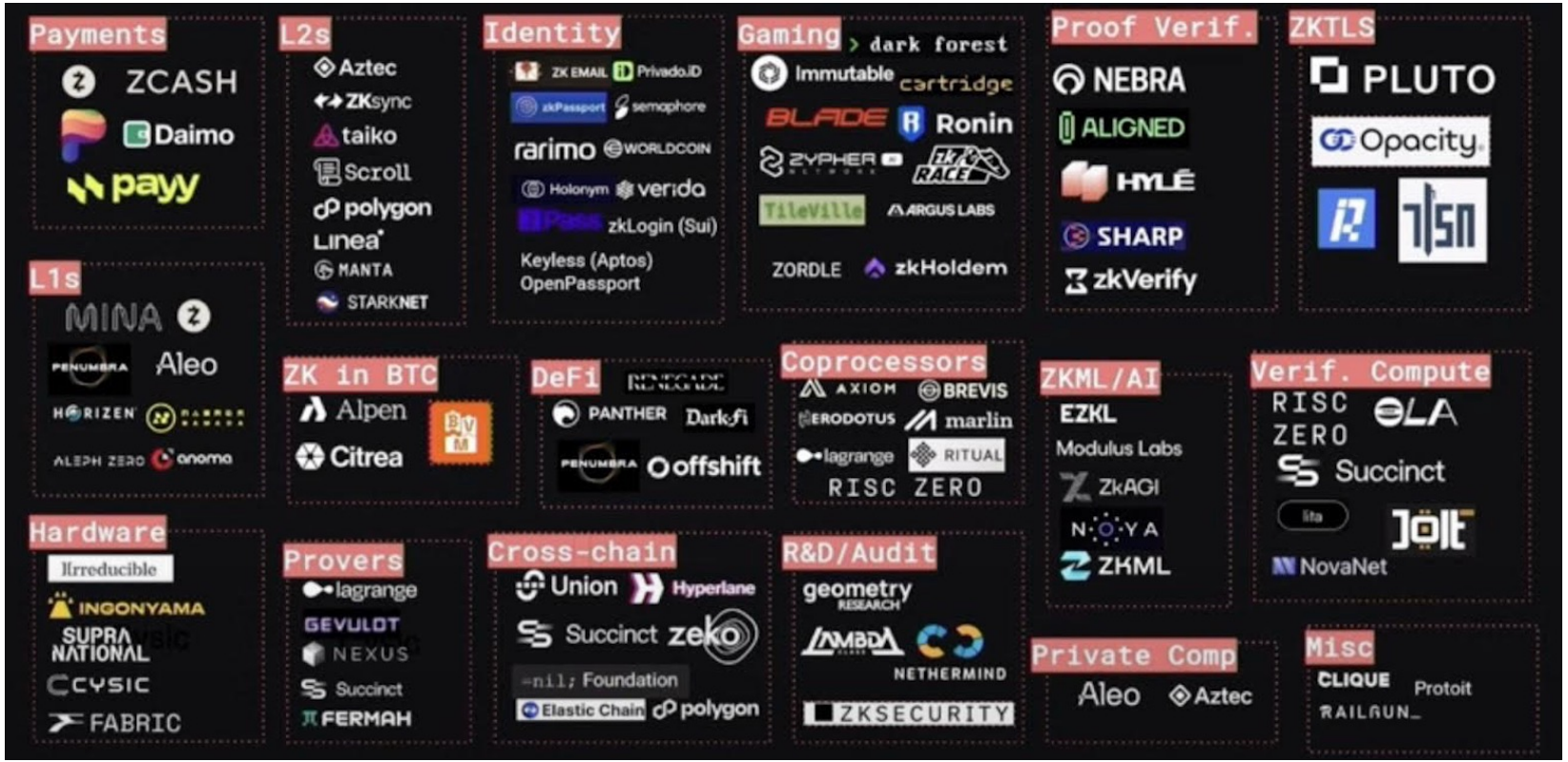
Example statement: "I know an m such that SHA256$(m) = 0$"

- **SNARK**: the proof is **"short"** and **"fast"** to verify

  [if m is 1GB then the trivial proof (the message m) is neither]

- **zk-SNARK**: the proof "reveals nothing" about m

Commercial interest in SNARKs



Many more companies using (zk)-SNARKs: https://zkhack.dev/the-map-of-zk/

Blockchain Applications I

Outsourcing computation: (no need for zero knowledge)

L1 chain quickly verifies the work of an off-chain service

To minimize gas: need a short proof, fast to verify

Examples:

- **Scalability:** proof-based Rollups (zkRollup)

  off-chain service processes a batch of Tx;

  L1 chain verifies a succinct proof that Tx were processed correctly

- **Bridging blockchains:** proof of consensus (zkBridge)

  Chain A produces a succinct proof about its state. Chain B verifies.

Blockchain Applications II

Some applications require zero knowledge (privacy):

- **Private Tx on a public blockchain**:
  - zk proof that a private Tx is valid (Tornado cash, Railgun, Zcash, Aleo)
- **Compliance:**
  - Proof that a private Tx is compliant with banking laws
  - Proof that an exchange is solvent in zero-knowledge (Proven)

More on these blockchain applications in a minute

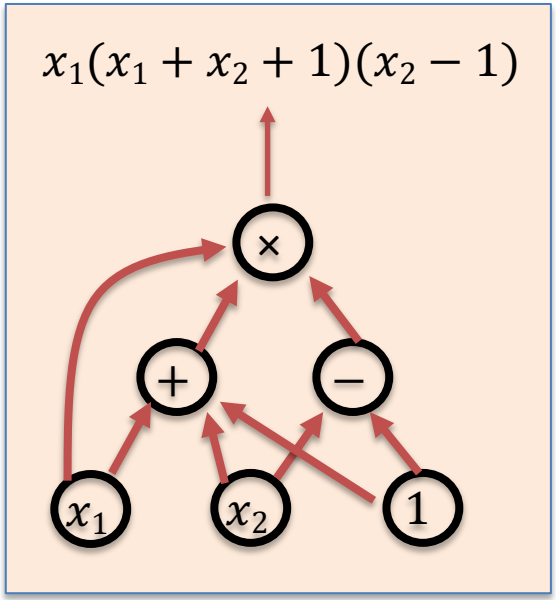
Many non-blockchain applications

Blockchains drive the development of SNARKs

… but **many** non-blockchain applications benefit

Arithmetic circuits

- Fix a finite field $\mathbb{F} = \{0, \ldots, p-1\}$ for some prime p>2.
- **Arithmetic circuit**: $C: \mathbb{F}^n \to \mathbb{F}$
  - directed acyclic graph (DAG) where
    internal nodes are labeled +, −, or ×
    inputs are labeled $1, x_1, \ldots, x_n$
  - defines an n-variate polynomial with an evaluation recipe
- $|C|$ = # gates in C



Interesting arithmetic circuits

Examples:

- $C_{hash}(h, \mathbf{m})$: outputs 0 if SHA256($\mathbf{m}$) = h , and ≠0 otherwise

  $C_{hash}(h, \mathbf{m}) = (h - \text{SHA256}(\mathbf{m}))$, $|C_{hash}| \approx 20K$ gates

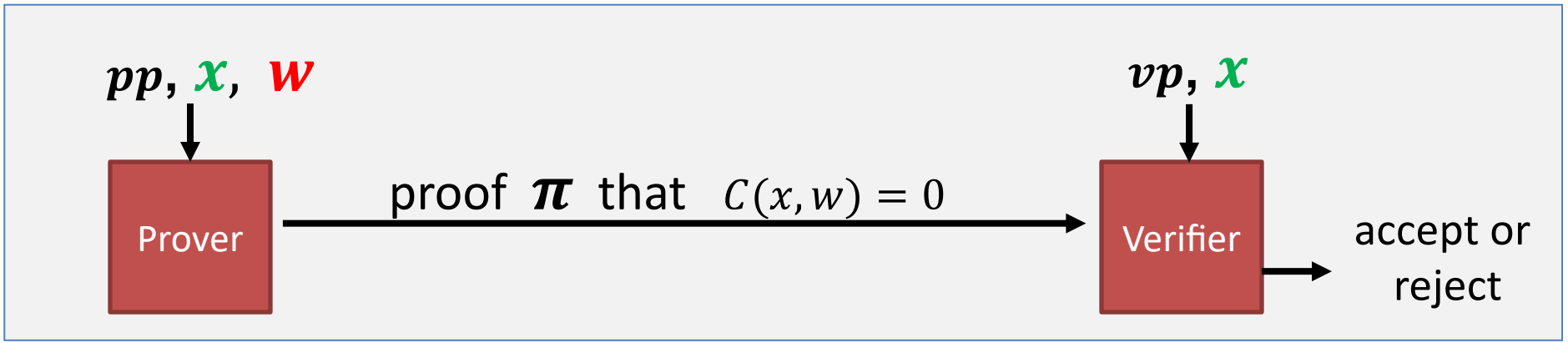
- $C_{sig}(pk, m, \sigma)$: outputs 0 if σ is a valid ECDSA signature on m with respect to pk

(preprocessing) NARK: Non-interactive ARgument of Knowledge

Public arithmetic circuit: $C(\boldsymbol{x}, \boldsymbol{w}) \rightarrow \mathbb{F}$

public statement in $\mathbb{F}^n$ — secret witness in $\mathbb{F}^m$

Preprocessing (setup): $\mathbf{S}(C) \rightarrow$ public parameters $(\boldsymbol{pp}, \boldsymbol{vp})$



(preprocessing) NARK: Non-interactive ARgument of Knowledge

A **preprocessing NARK** is a triple (S, P, V):

- **S**$(C) \rightarrow$ public parameters (pp, vp) for prover and verifier
- **P**$(pp, \boldsymbol{x}, \boldsymbol{w}) \rightarrow$ proof π
- **V**$(vp, \boldsymbol{x}, \boldsymbol{\pi}) \rightarrow$ accept or reject

NARK: requirements (informal)

Prover P(pp, $\boldsymbol{x}$, $\boldsymbol{w}$) — Verifier V (vp, $\boldsymbol{x}$, $\boldsymbol{\pi}$)

proof π →

accept or reject

Complete: $\forall x, w$: $C(\boldsymbol{x}, \boldsymbol{w}) = 0 \Rightarrow$ Pr[V(vp, x, P(pp, $\boldsymbol{x}$, $\boldsymbol{w}$)) = accept] = 1

knowledge sound: V accepts $\Rightarrow$ P "knows" $\boldsymbol{w}$ s.t. $C(\boldsymbol{x}, \boldsymbol{w}) = 0$

(an extractor E can extract a valid $\boldsymbol{w}$ from P)

Optional: **Zero knowledge**: $(C, pp, vp, \boldsymbol{x}, \pi)$ "reveal nothing" about $\boldsymbol{w}$

SNARK: a Succinct ARgument of Knowledge

A **succinct preprocessing NARK** is a triple (S, P, V):

- **S**$(C) \rightarrow$ public parameters (pp, vp) for prover and verifier
- **P**$(pp, \boldsymbol{x}, \boldsymbol{w}) \rightarrow$ **short** proof π ; $\text{len}(\pi) = O_\lambda(\textbf{polylog}(|\boldsymbol{C}|))$
- **V**$(vp, \boldsymbol{x}, \boldsymbol{\pi})$ **fast to verify** ; $\text{time(V)} = O_\lambda(|x|, \textbf{polylog}(|\boldsymbol{C}|))$

short "summary" of circuit

V has no time to read C !!

[for some SNARKs, $\text{len}(\pi) = \text{time}(V) = O_\lambda(1)$]

SNARK: a Succinct ARgument of Knowledge

SNARK: a NARC (complete and knowledge sound) that is **succinct**

zk-SNARK: a SNARK that is also **zero knowledge**

The trivial SNARK is not a SNARK

(a) Prover sends $\boldsymbol{w}$ to verifier,

(b) Verifier checks if $C(\boldsymbol{x}, \boldsymbol{w}) = 0$ and accepts if so.

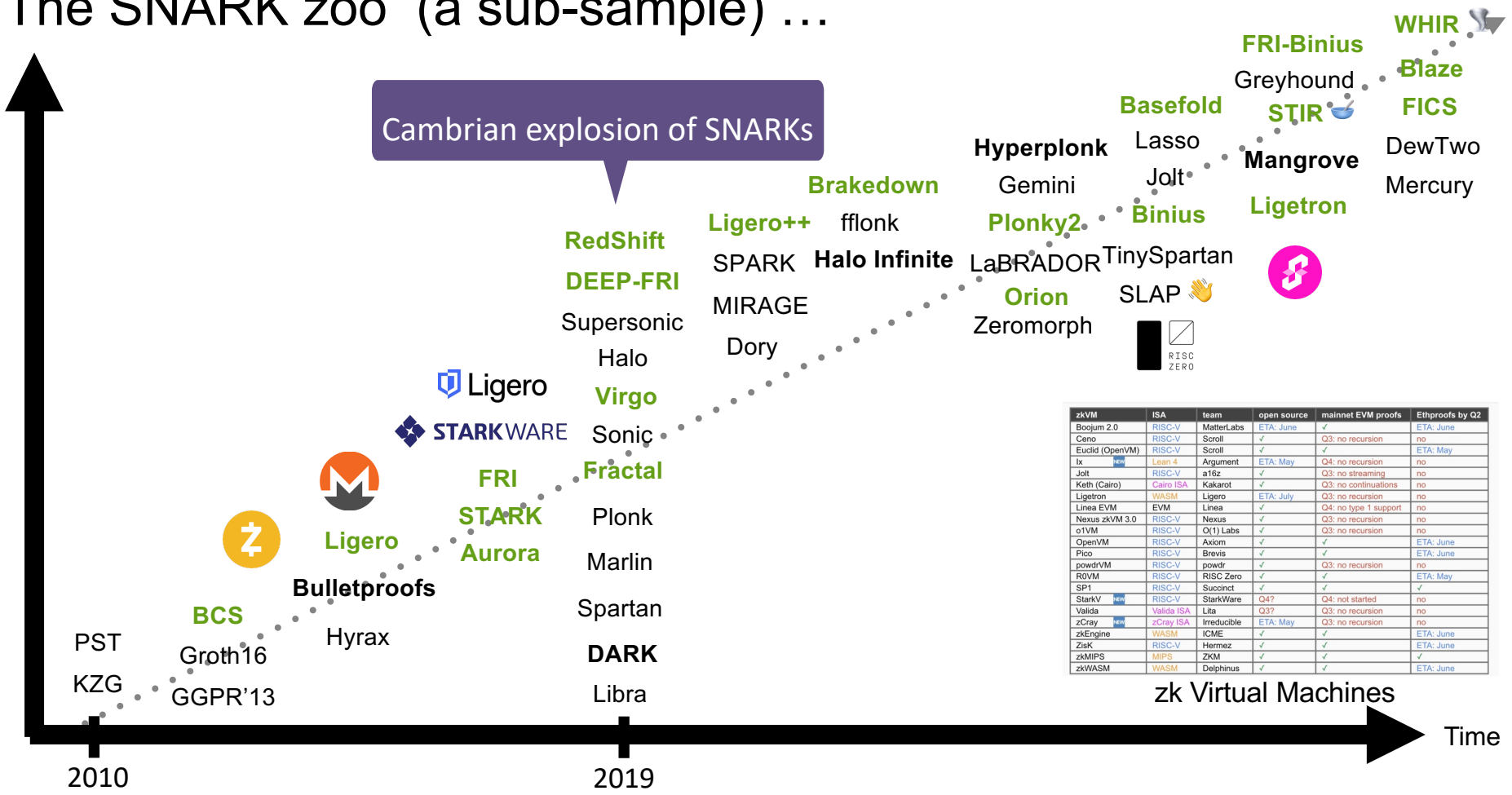
Problems with this:

(1) $\boldsymbol{w}$ might be long: we want a "short" proof

(2) computing $C(\boldsymbol{x}, \boldsymbol{w})$ may be hard: we want a "fast" verifier

(3) $\boldsymbol{w}$ might be secret: prover might not want to reveal $\boldsymbol{w}$ to verifier

The SNARK zoo (a sub-sample) …

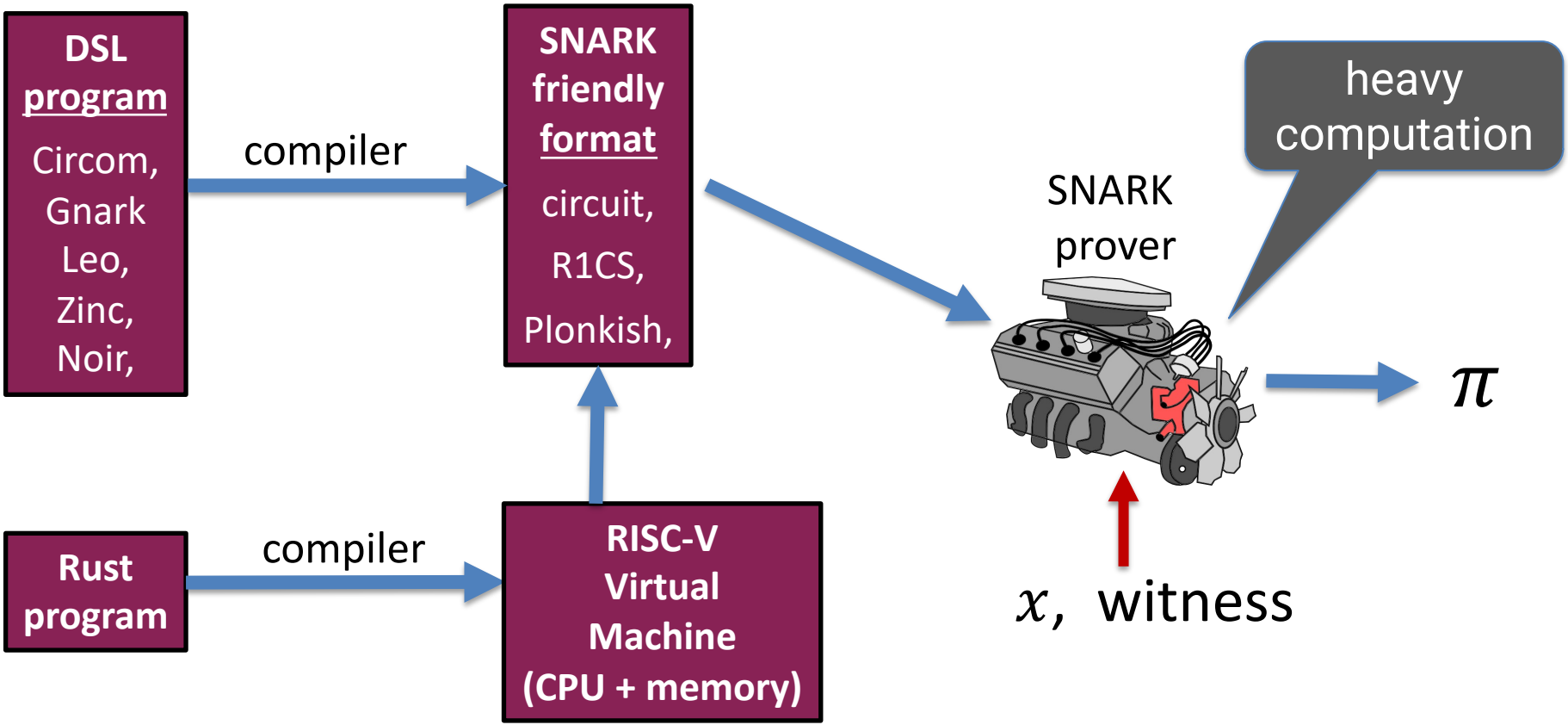


zkVM	ISA	team	open source	mainnet EVM proofs	Ethproofs by Q2
Boojum 2.0	RISC-V	MatterLabs	ETA: June	✓	ETA: June
Ceno	RISC-V	Scroll	✓	Q3: no recursion	no
Euclid (OpenVM)	RISC-V	Scroll	✓	✓	ETA: May
Ix NEW	Lean 4	Argument	ETA: May	Q4: no recursion	no
Jolt	RISC-V	a16z	✓	Q3: no streaming	no
Keth (Cairo)	Cairo ISA	Kakarot	✓	Q3: no continuations	no
Ligetron	WASM	Ligero	ETA: July	Q3: no recursion	no
Linea EVM	EVM	Linea	✓	Q4: no type 1 support	no
Nexus zkVM 3.0	RISC-V	Nexus	✓	Q3: no recursion	no
o1VM	RISC-V	O(1) Labs	✓	Q3: no recursion	no
OpenVM	RISC-V	Axiom	✓	✓	ETA: June
Pico	RISC-V	Brevis	✓	✓	ETA: June
powdrVM	RISC-V	powdr	✓	Q3: no recursion	no
R0VM	RISC-V	RISC Zero	✓	✓	ETA: May
SP1	RISC-V	Succinct	✓	✓	✓
StarkV NEW	RISC-V	StarkWare	Q4?	Q4: not started	no
Valida	Valida ISA	Lita	Q3?	Q3: no recursion	no
zCray NEW	zCray ISA	Irreducible	ETA: May	Q3: no recursion	no
zkEngine	WASM	ICME	✓	✓	ETA: June
ZisK	RISC-V	Hermez	✓	✓	ETA: June
zkMIPS	MIPS	ZKM	✓	✓	✓
zkWASM	WASM	Delphinus	✓	✓	ETA: June

zk Virtual Machines

Credit: Giacomo Fenzi

SNARKs in practice: two approaches



END OF LECTURE

Next lecture:
more on zk-SNARKs and their applications