STARKWARE

ZK-STARK
Theory & Implementation

Eli Ben-Sasson / Co-Founder & President

@elibensasson | @starkwareltd

November 2021

Overview

1. My story and “red pill” moment
2. The Cambrian Explosion of ZKPs
3. ZK-STARKs unleashed
4. How to build a STARK?
5. [Fast RS IOPPs (FRI)] *time permitting*

My Story

My Research and Blockchain

- 2001: Postdoc at Harvard+MIT, Madhu Sudan suggested studying PCP length
- 2003-5: Short PCPs with poly-log query complexity [BS04]
  - Theoretical result, no practical application in sight
- 2008: Students start implementing it in code
  - Why? No clear reason
- 2009: Huge ERC funding (1.7M Euro), more implementation
  - Why? Still no good reason
- 2013: Bitcoin San Jose Conference
  - Red pill swallowed
  - Why?

Post Red Pill

- 2014: Zerocash academic paper
- 2015: Zcash launched
- 2013-16: Startup failed attempt
- 2018: Math breakthroughs, not well-received
  - FRI: Rejected from 3 conferences (including STOC/FOCS and ITCS, accepted to ICALP)
  - STARK: Rejected from 4 conferences (including CRYPTO, CCS, accepted to CRYPTO)
  - PCP Security: Rej from 3 conferences (gave up)

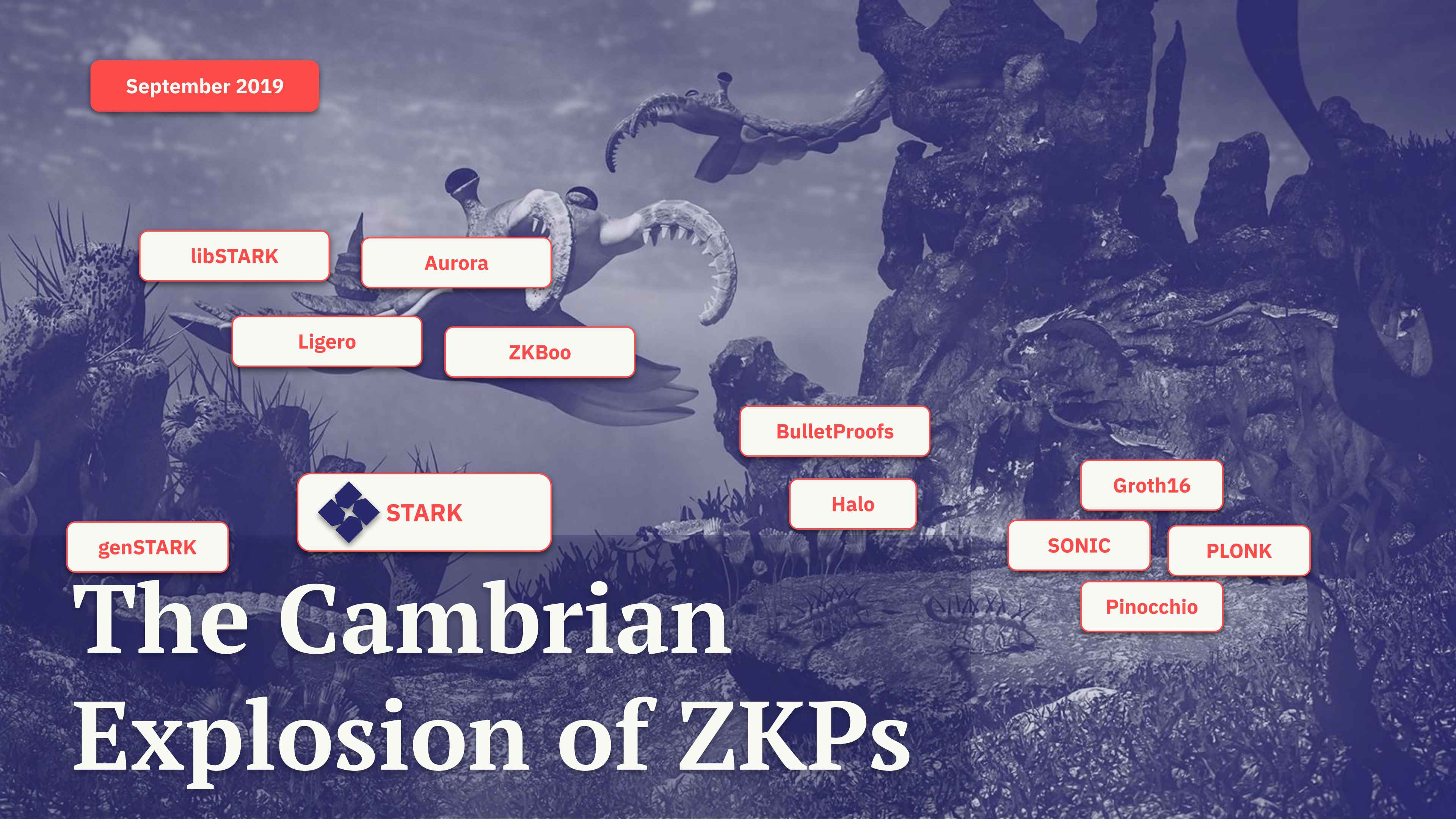
Meanwhile in Blockchain world…

- Zcash=> ZKP/ ZK-SNARKs hype
- Huge enthusiasm for ZK-STARKs
- 2018: StarkWare Founded
  - My co-founders: Alessandro Chiesa, Uri Kololdny, Michael Riabzev
  - $6M funding, followed by $25M, …
- At launch, still missing:
  - key math results: DEEP FRI, tight soundness analysis, …
  - Accessibility: Cairo language, system, business model, product …
  - But we knew very well what we’ll do

Overview

1. My story and “red pill” moment
2. The Cambrian Explosion of ZKPs
3. ZK-STARKs unleashed
4. How to build a STARK?
5. [Fast RS IOPPs (FRI)] *time permitting*

September 2019



libSTARK

Aurora

Ligero

ZKBoo

BulletProofs

STARK

Halo

Groth16

genSTARK

SONIC

PLONK

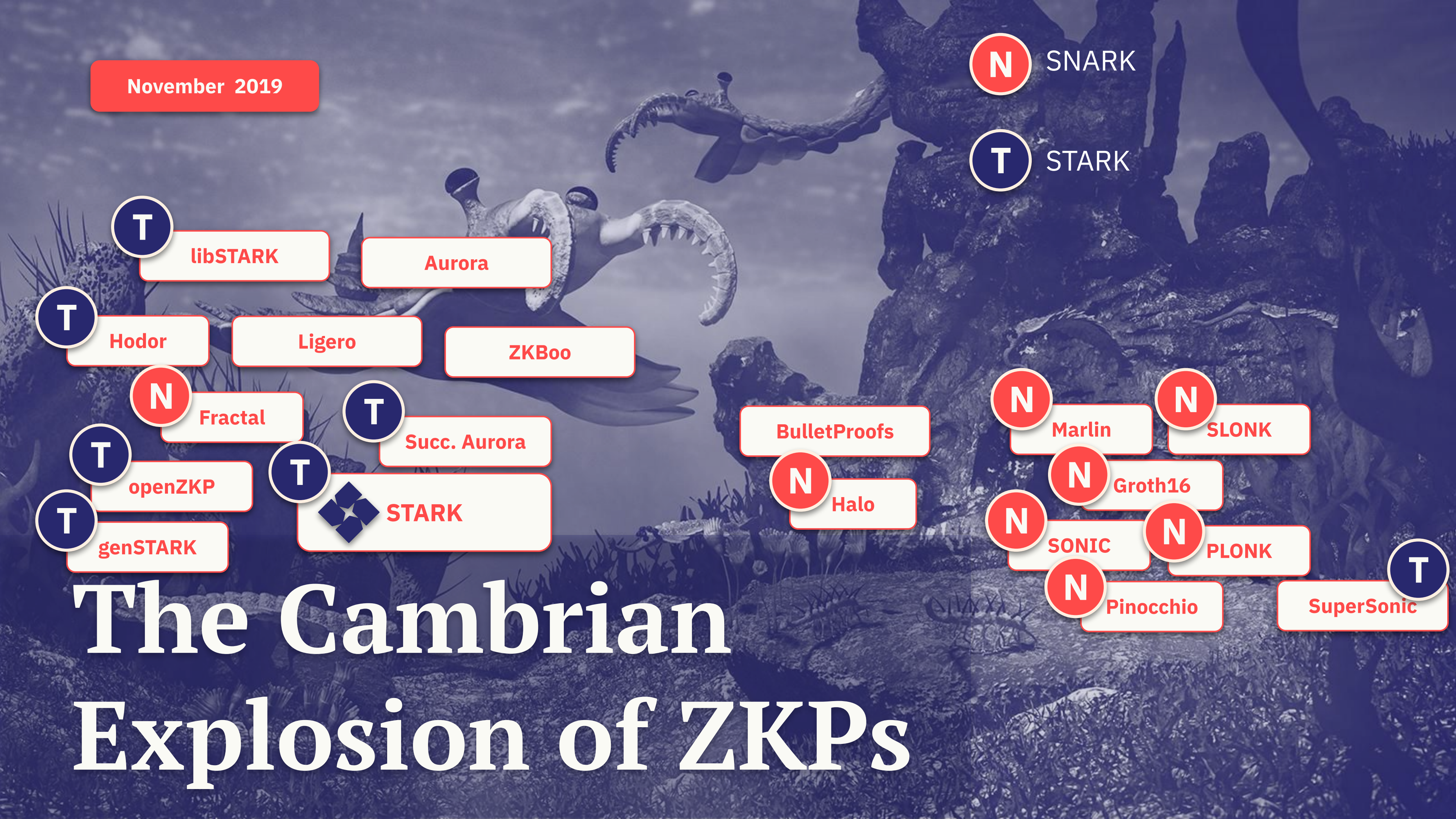
Pinocchio

The Cambrian Explosion of ZKPs

November 2019

N SNARK

T STARK



T libSTARK

Aurora

T Hodor

Ligero

ZKBoo

N Fractal

T Succ. Aurora

T openZKP

T STARK

T genSTARK

BulletProofs

N Halo

N Marlin

N SLONK

N Groth16

N SONIC

N PLONK

N Pinocchio

T SuperSonic

The Cambrian Explosion of ZKPs

Proofs of Computational Integrity (CI)

Privacy (Zero Knowledge, ZK)
Prover’s private inputs are shielded

Scalability
Exponentially small verifier running time*
Nearly linear prover running time*

Universality
Applicability to general computation

Transparency
No toxic waste (i.e. no trusted setup)

Lean & Battle-Hardened Cryptography
e.g. post-quantum secure

STARK

*With respect to size of computation

STARKWARE

Proofs of Computational Integrity (CI)

Privacy (Zero Knowledge, ZK)
Prover's private inputs are shielded

Scalability
Exponentially small verifier running time*
Nearly linear prover running time*

Universality
Applicability to general computation

Transparency
No toxic waste (i.e. no trusted setup)

Lean & Battle-Hardened Cryptography
e.g. post-quantum secure

(ZK)-STARK

*With respect to size of computation

STARK vs. SNARK - emphasizing different aspects

T STARKs *must be*

Transparent no trusted setup

Scalable: logarithmic verifying time **and** nearly-linear proving time

Succinct setup, at most logarithmic time

N SNARKs *must be*

Noninteractive: pf is single message (after preprocessing)

Succinct: logarithmic verifying time

Setup can take linear time (and more)

Non-interactive STARKs are SNARKs (transparent ones)

Transparent SNARKs w/ succinct setup are STARKs

libSTARK

Aurora

Hodor

Ligero

ZKBoo

Fractal

Succ. Aurora

openZKP

STARK

genSTARK

BulletProofs

Halo

Marlin

SLONK

Groth16

SONIC

PLONK

Pinocchio

SuperSonic

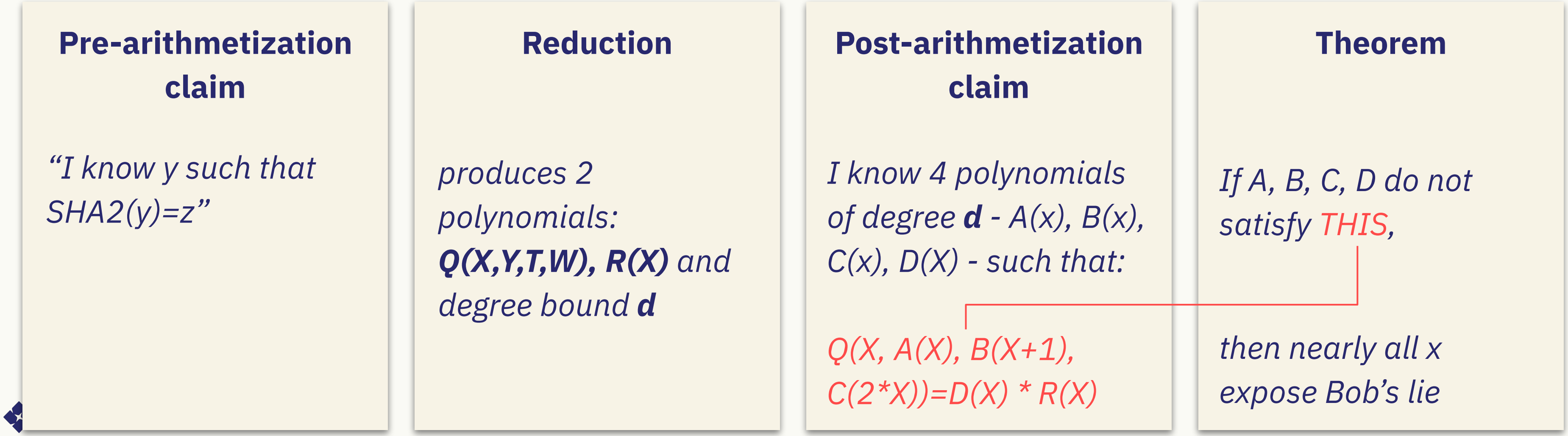
Common Ancestors

1. Arithmetization
2. Low degreeness

1) Arithmetization

Arithmetization Converts (“reduces”) Computational Integrity problems to problems about local relations between a bunch of polynomials

Example: For public 256-bit string ***z***, Bob claims knows a SHA2-preimage of ***z***



Pre-arithmetization claim

“I know y such that SHA2(y)=z”

Reduction

*produces 2 polynomials: **Q(X,Y,T,W), R(X)** and degree bound **d***

Post-arithmetization claim

*I know 4 polynomials of degree **d** - A(x), B(x), C(x), D(X) - such that:*

*Q(X, A(X), B(X+1), C(2*X))=D(X) * R(X)*

Theorem

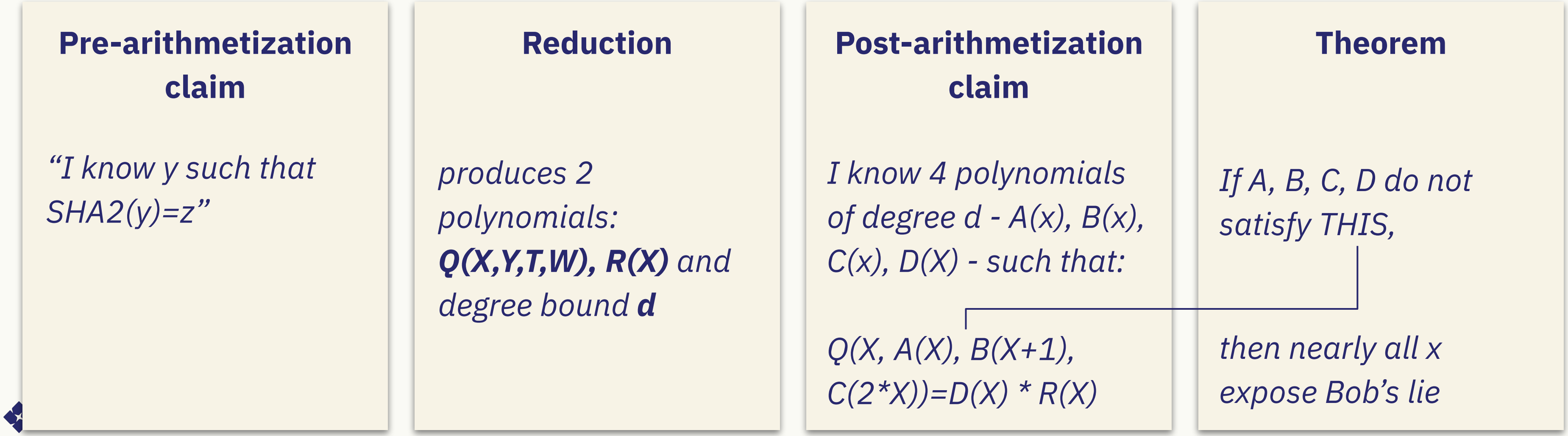
If A, B, C, D do not satisfy THIS,

then nearly all x expose Bob’s lie

1) Arithmetization

Assuming Theorem, we get a scalable proof system for Bob’s original claim:

1. Apply reduction, ask Bob to provide access to A,B,C,D of degree-d
2. Sample random x and accept Bob’s claim iff equality holds for this x



Pre-arithmetization claim

“I know y such that SHA2(y)=z”

Reduction

produces 2 polynomials: ***Q(X,Y,T,W), R(X)*** *and degree bound* ***d***

Post-arithmetization claim

I know 4 polynomials of degree d - A(x), B(x), C(x), D(X) - such that:

$Q(X, A(X), B(X+1), C(2*X))=D(X) * R(X)$

Theorem

If A, B, C, D do not satisfy THIS,

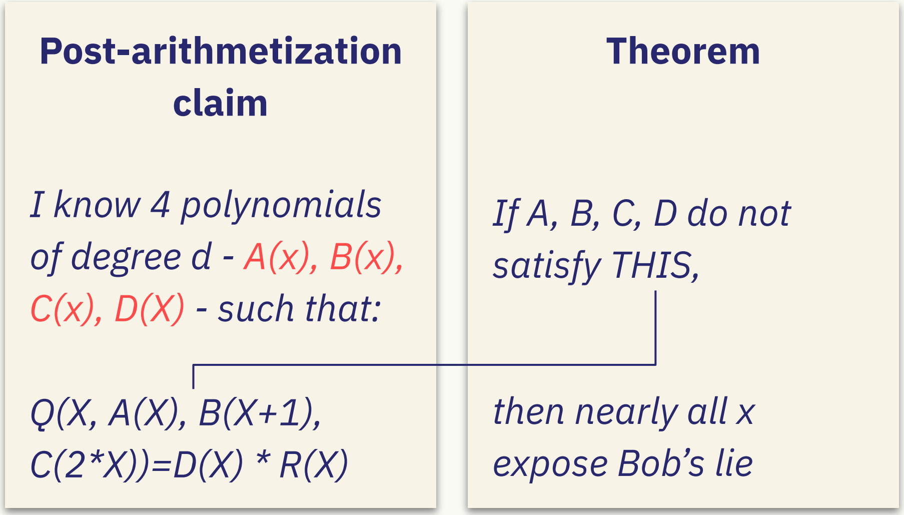
then nearly all x expose Bob’s lie

2) Low degreeness

Assuming Theorem, we get a scalable proof system for Bob's original claim:

1. Apply reduction, ask Bob to provide access to A,B,C,D of degree-d
2. Sample random x and accept Bob's claim iff equality holds for this x

New Computational Integrity problem: Force Bob to answer all queries according to some quadruple of degree-d polynomials



Post-arithmetization claim

I know 4 polynomials of degree d - A(x), B(x), C(x), D(X) - such that:

$Q(X, A(X), B(X+1), C(2*X))=D(X) * R(X)$

Theorem

If A, B, C, D do not satisfy THIS,

then nearly all x expose Bob's lie

Differentiating Factors

1. Arithmetization Method
2. Low degreeness enforcement
3. Cryptographic assumptions used to get 2

libSTARK

Aurora

Hodor

Ligero

ZKBoo

Fractal

Succ. Aurora

openZKP

STARK

genSTARK

BulletProofs

Halo

Marlin

SLONK

Groth16

SONIC

PLONK

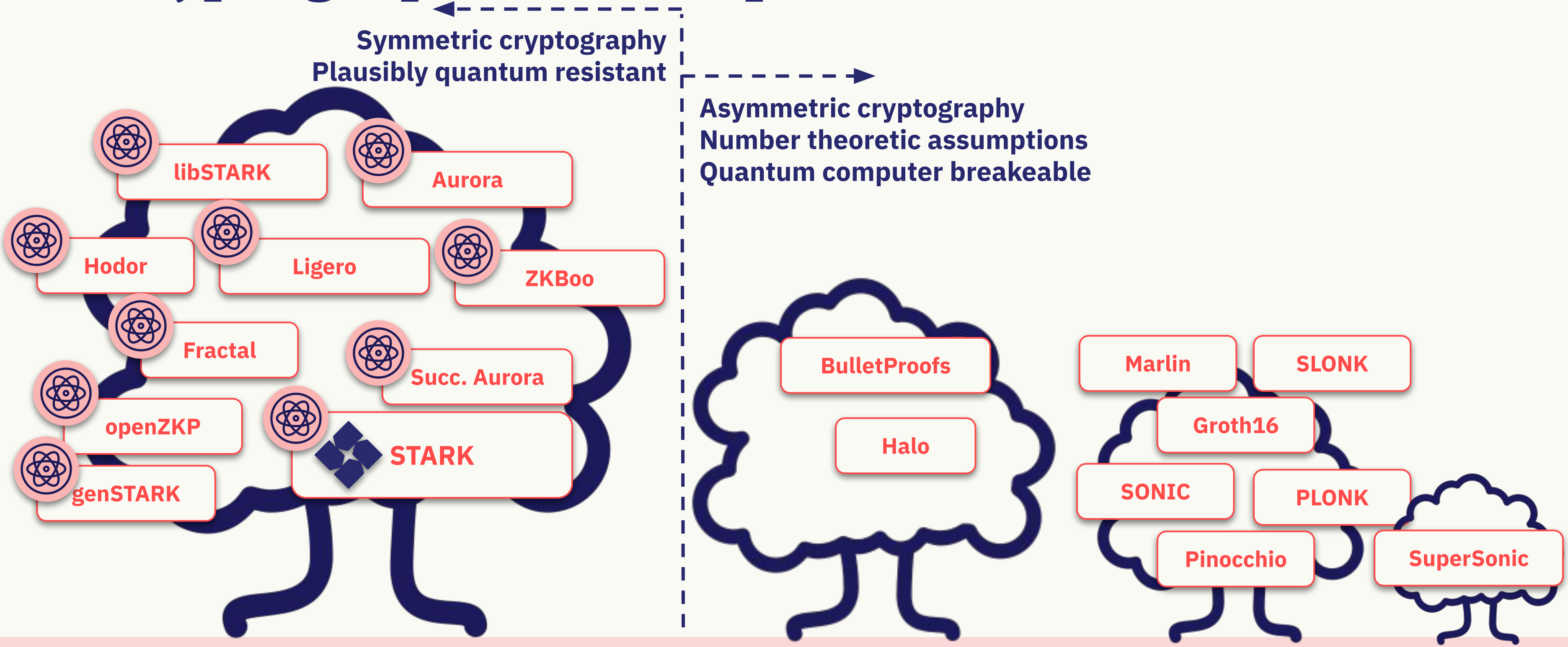
Pinocchio

SuperSonic

Common Ancestors

1. Arithmetization
2. Low degreeness

3. Cryptographic Assumptions



Symmetric cryptography
Plausibly quantum resistant

Asymmetric cryptography
Number theoretic assumptions
Quantum computer breakeable

libSTARK

Aurora

Hodor

Ligero

ZKBoo

Fractal

Succ. Aurora

openZKP

STARK

genSTARK

BulletProofs

Halo

Marlin

SLONK

Groth16

SONIC

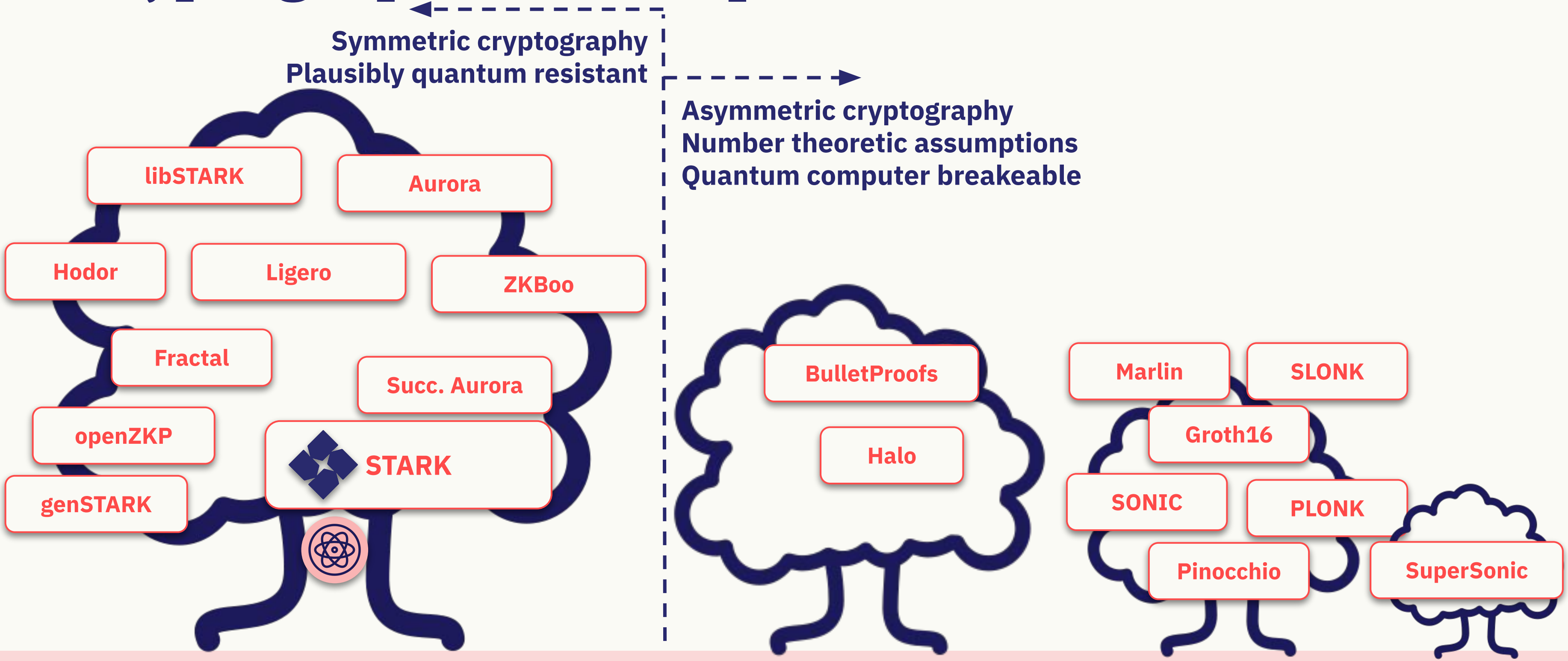
PLONK

Pinocchio

SuperSonic

Cryptographic Assumptions	Collision-Resistant Hash	Elliptic Curve DLP	Knowledge of Exponent	Groups of unknown order
year	1976	1980s-2000s	2000s-2017	1997-2019

3. Cryptographic Assumptions

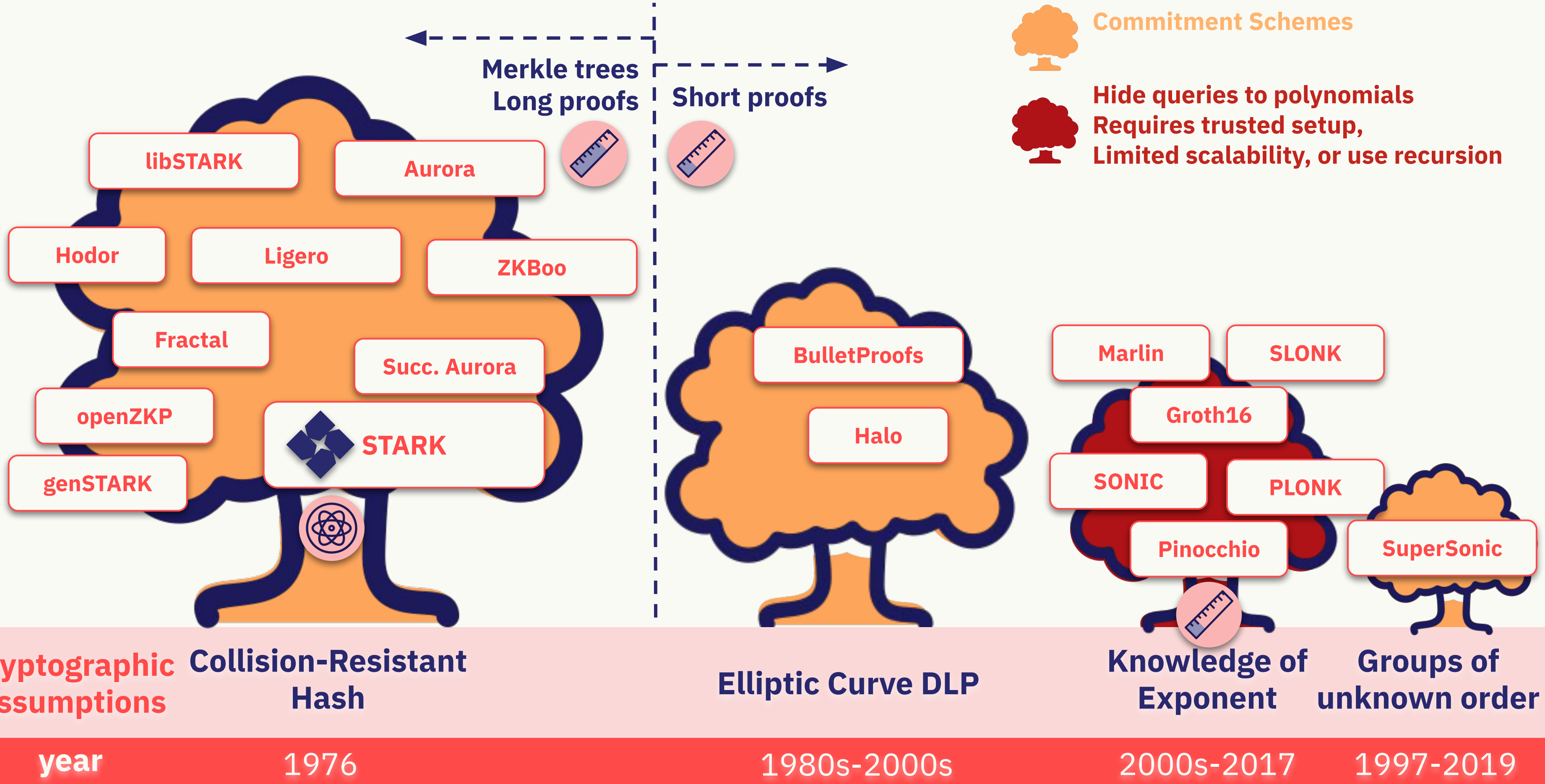


Symmetric cryptography
Plausibly quantum resistant

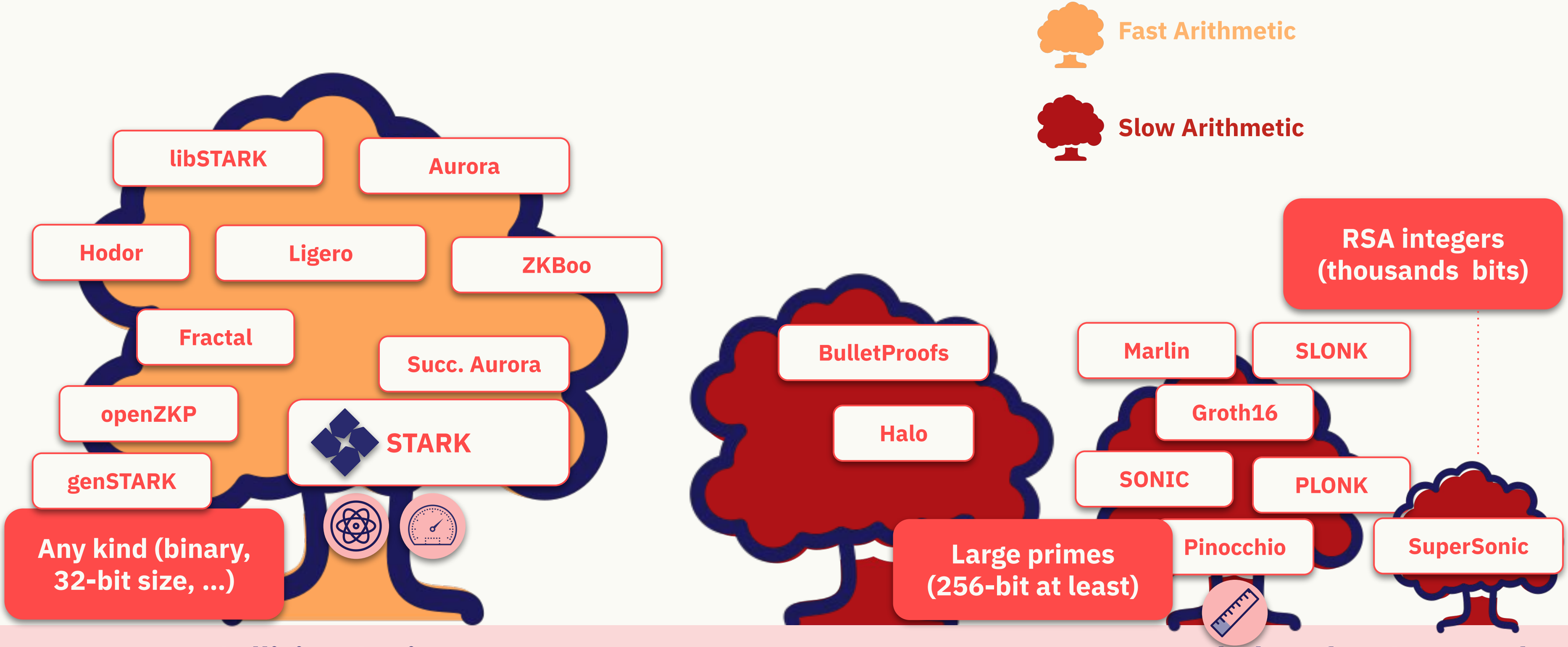
Asymmetric cryptography
Number theoretic assumptions
Quantum computer breakeable

Cryptographic Assumptions	Collision-Resistant Hash	Elliptic Curve DLP	Knowledge of Exponent	Groups of unknown order
year	1976	1980s-2000s	2000s-2017	1997-2019

2. Enforcing low-degreeness



1. Arithmetization - finite field type



Fast Arithmetic

Slow Arithmetic

libSTARK

Aurora

Hodor

Ligero

ZKBoo

Fractal

Succ. Aurora

openZKP

STARK

genSTARK

Any kind (binary, 32-bit size, ...)

BulletProofs

Halo

Large primes (256-bit at least)

Marlin

SLONK

Groth16

SONIC

PLONK

Pinocchio

RSA integers (thousands bits)

SuperSonic

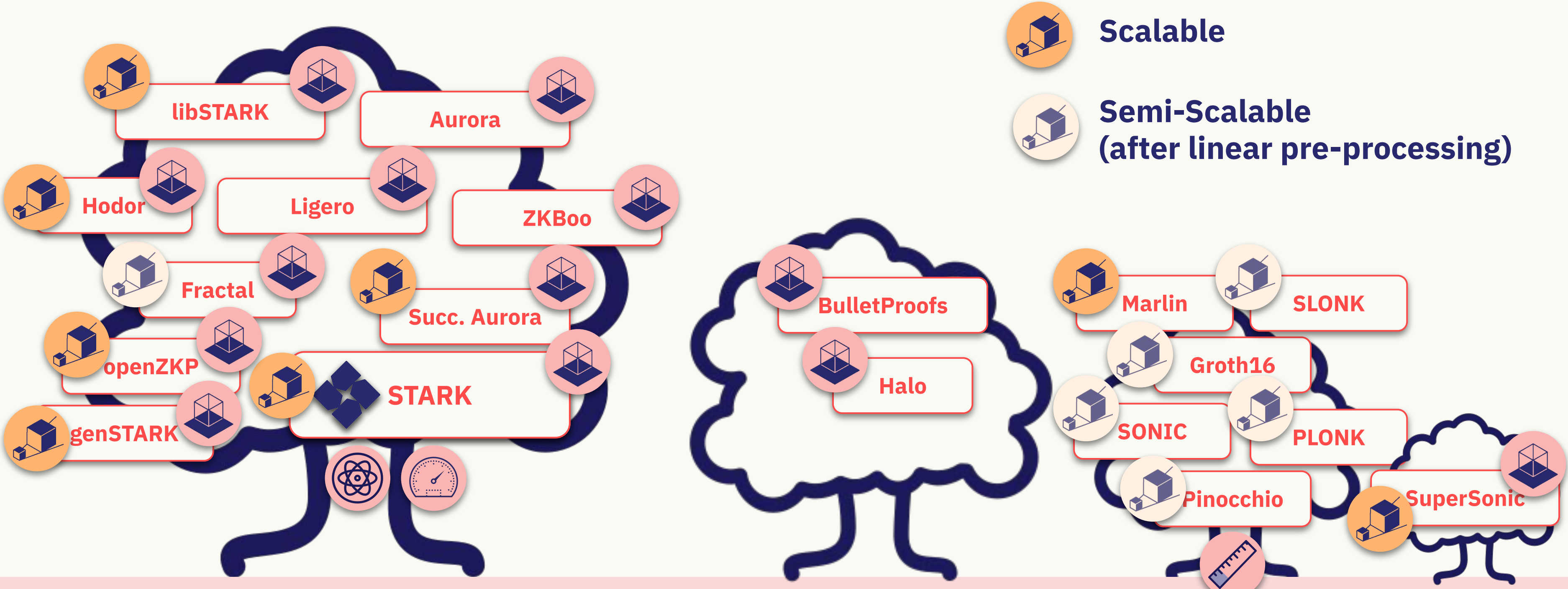
Cryptographic Assumptions	Collision-Resistant Hash	Elliptic Curve DLP	Knowledge of Exponent	Groups of unknown order
year	1976	1980s-2000s	2000s-2017	1997-2019

Scalability and Transparency

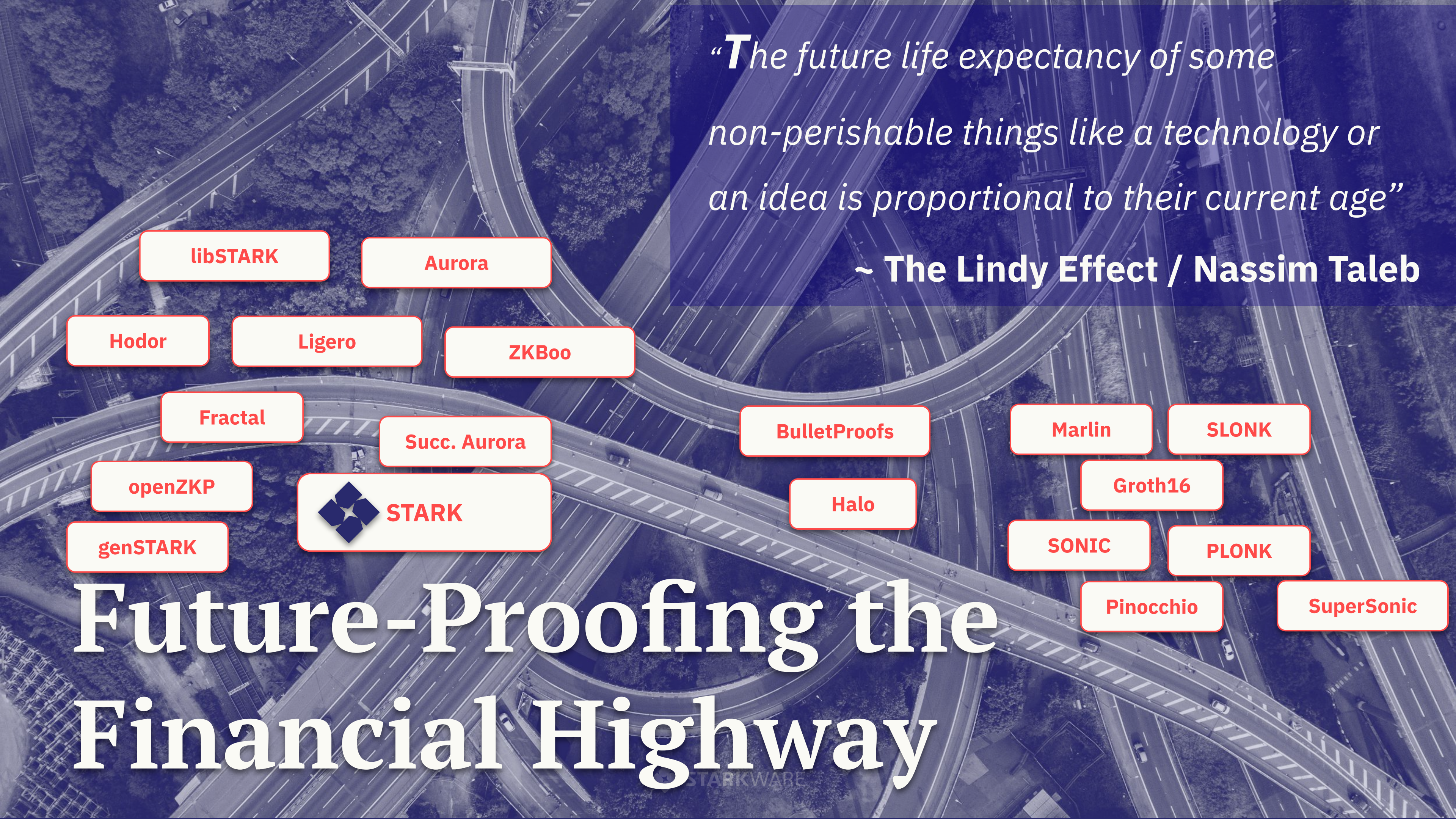
Transparent

Scalable

Semi-Scalable
(after linear pre-processing)



Cryptographic Assumptions	Collision-Resistant Hash	Elliptic Curve DLP	Knowledge of Exponent	Groups of unknown order
year	1976	1980s-2000s	2000s-2017	1997-2019



“The future life expectancy of some non-perishable things like a technology or an idea is proportional to their current age”

~ The Lindy Effect / Nassim Taleb

Future-Proofing the Financial Highway

libSTARK

Aurora

Hodor

Ligero

ZKBoo

Fractal

Succ. Aurora

openZKP

STARK

genSTARK

BulletProofs

Halo

Marlin

SLONK

Groth16

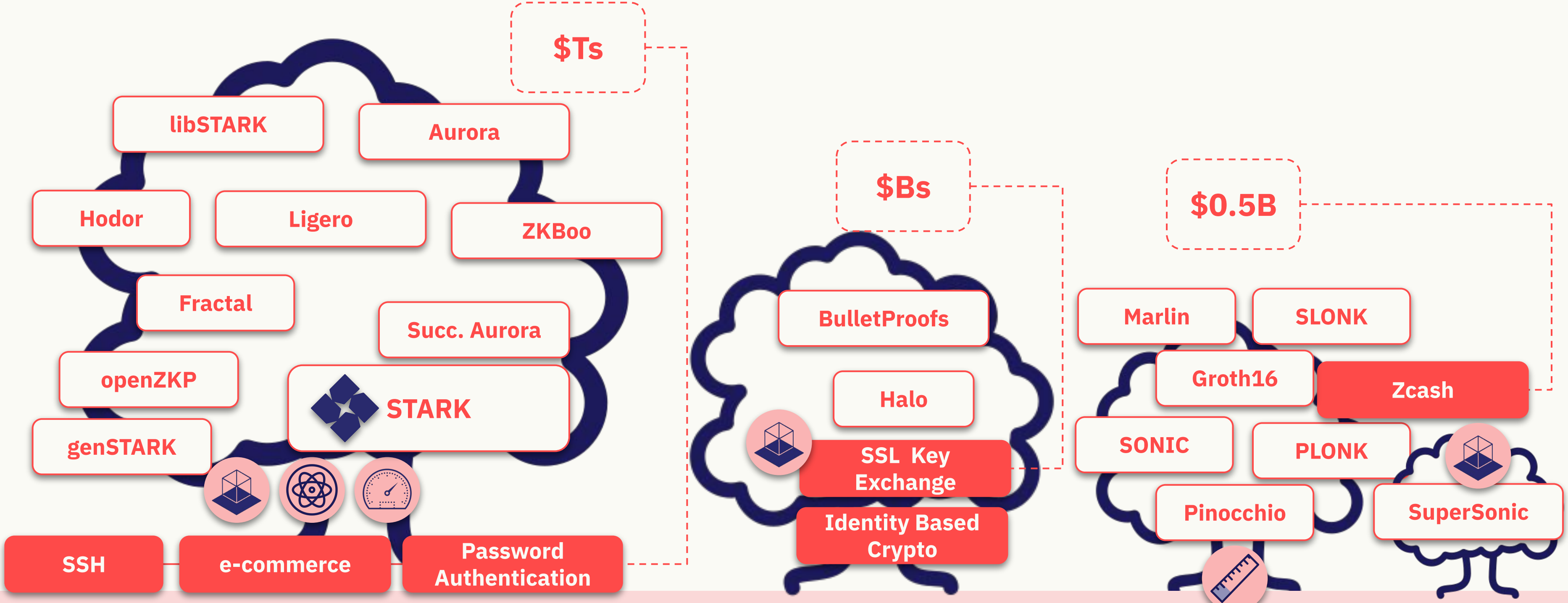
SONIC

PLONK

Pinocchio

SuperSonic

ZKP Family Trees



$Ts

libSTARK

Aurora

Hodor

Ligero

ZKBoo

Fractal

Succ. Aurora

openZKP

STARK

genSTARK

SSH

e-commerce

Password Authentication

$Bs

BulletProofs

Halo

SSL Key Exchange

Identity Based Crypto

$0.5B

Marlin

SLONK

Groth16

Zcash

SONIC

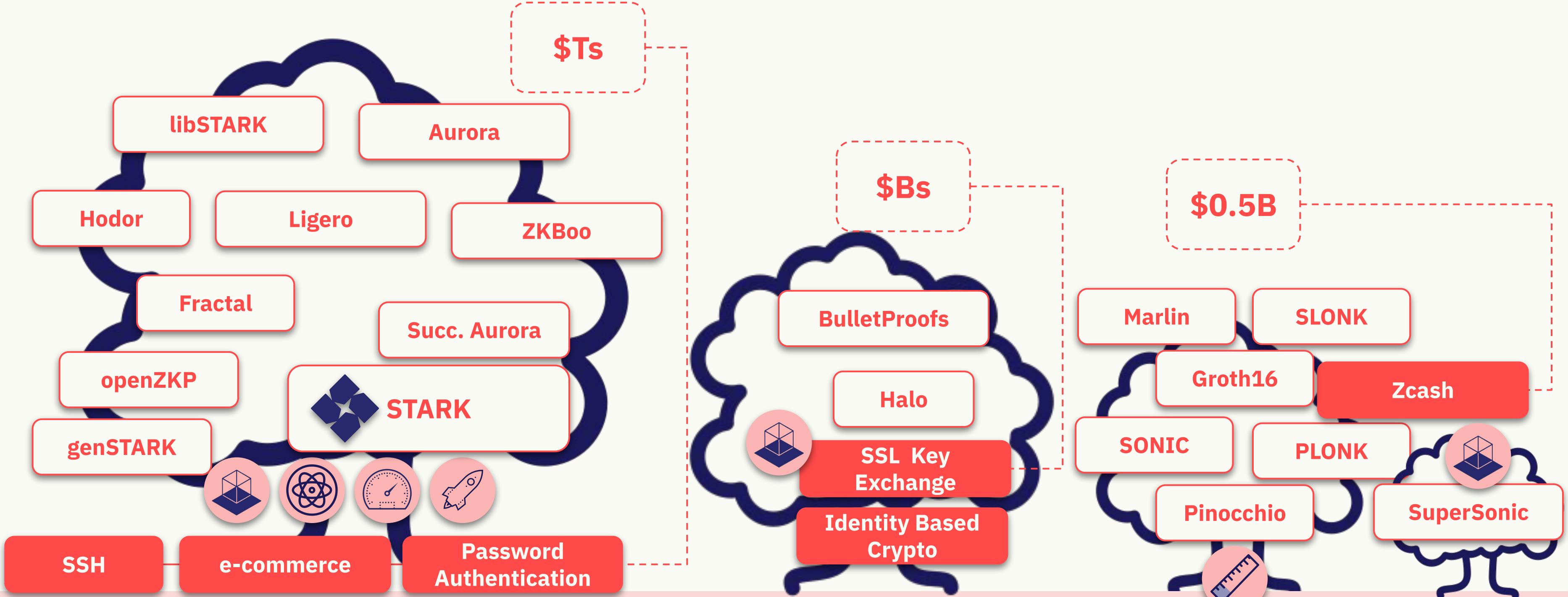
PLONK

Pinocchio

SuperSonic

Cryptographic Assumptions	Collision-Resistant Hash	Elliptic Curve DLP	Knowledge of Exponent	Groups of unknown order
year	1976	1980s-2000s	2000s-2017	1997-2019

ZKP Family Trees



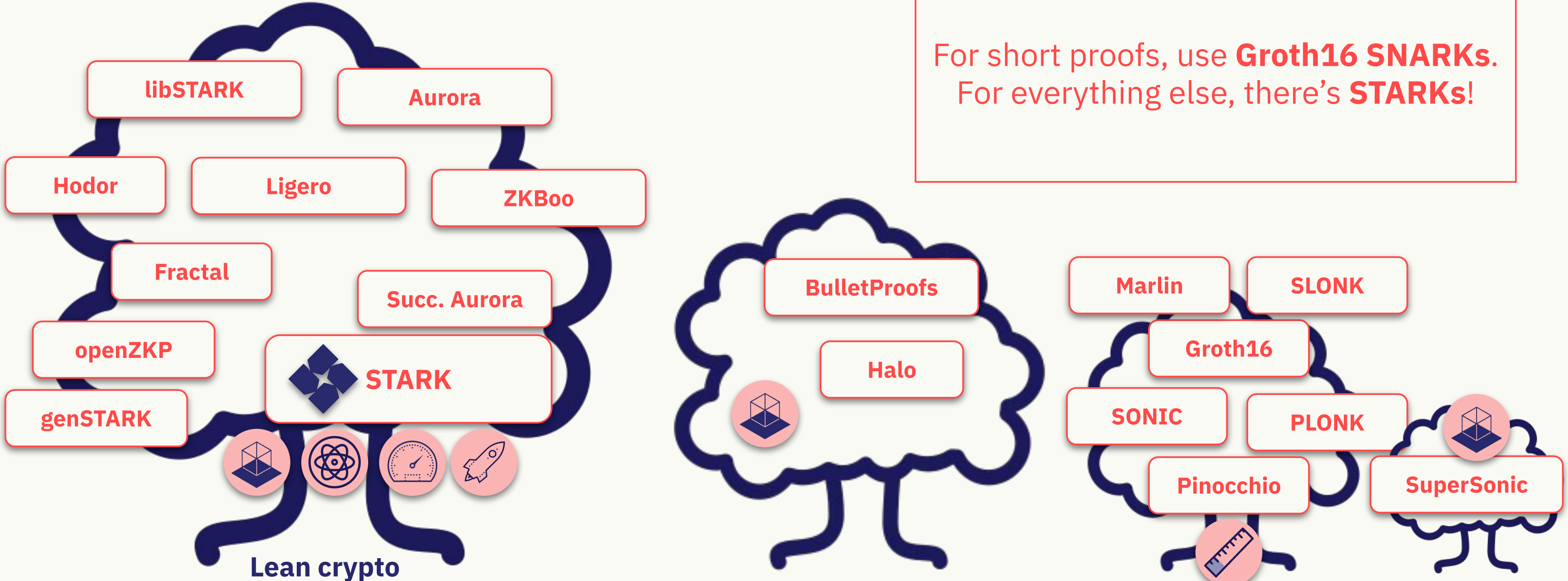
Cryptographic Assumptions	Collision-Resistant Hash	Elliptic Curve DLP	Knowledge of Exponent	Groups of unknown order
year	1976	1980s-2000s	2000s-2017	1997-2019

Summary

ZKP Cambrian explosion ongoing, expect more science!

ZKP members differ by (**i**) arithmetization, (**ii**) low-degreeness, and (**iii**) crypto assumptions

For short proofs, use **Groth16 SNARKs**.
For everything else, there's **STARKs**!



Lean crypto
Post quantum security
Fastest proving time
Future proofing (Lindsey)

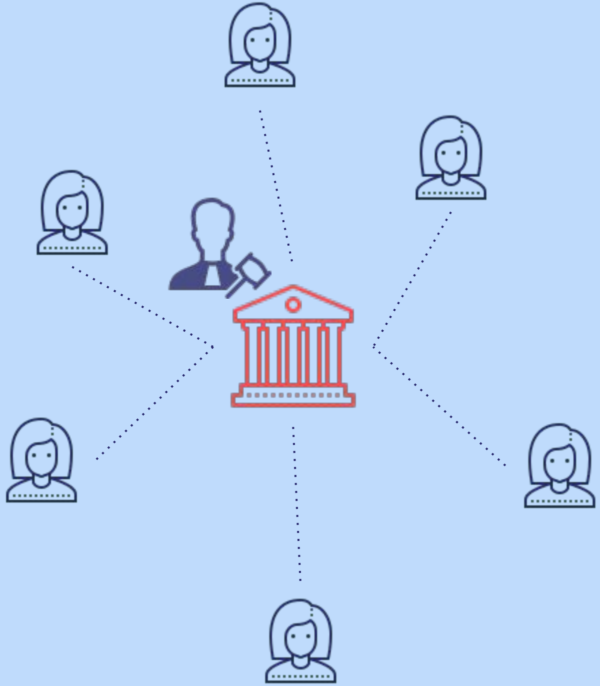
Proof length

Overview

1. My story and “red pill” moment
2. The Cambrian Explosion of ZKPs
3. ZK-STARKs unleashed
4. How to build a STARK?
5. [Fast RS IOPPs (FRI)] *time permitting*

Trusted Party (e.g., Banks) = Delegated Accountability

STARKWARE

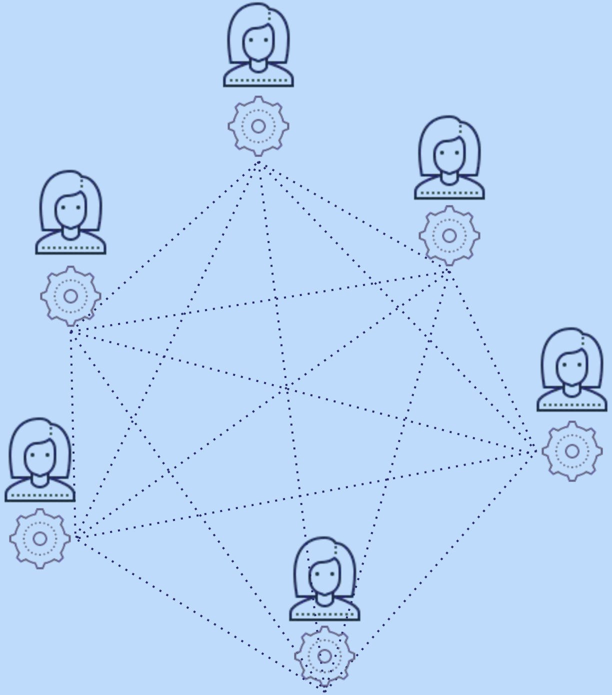


Trust central party/auditor

STARKWARE

Blockchains = Inclusive Accountability

Verify, Don’t Trust



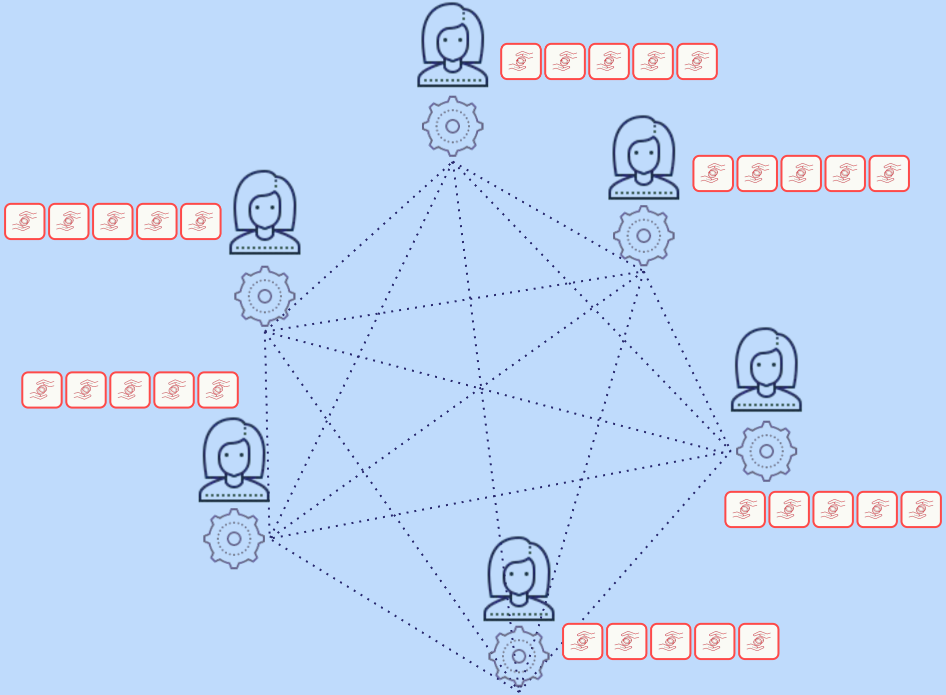
Verify (all transactions), don’t trust

Blockchains
=
Inclusive Accountability

Sacrifice Privacy & Scalability

ZK-STARKs
solve both problems

STARKWARE

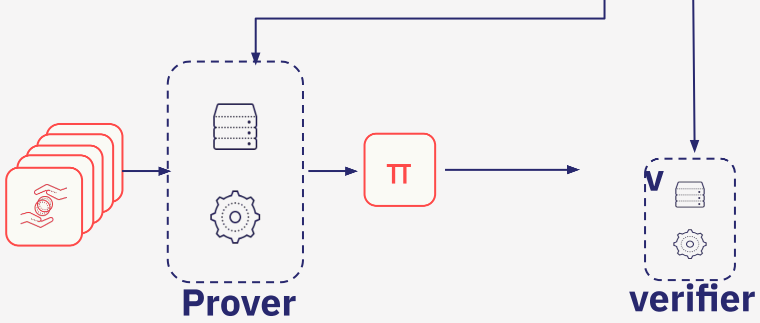


Verify (all transactions), don’t trust

ZK-STARK Proofs

Privacy (Zero Knowledge, ZK)
Prover’s private inputs are shielded

Scalability
Exponentially small verifier running time*
Nearly linear prover running time*

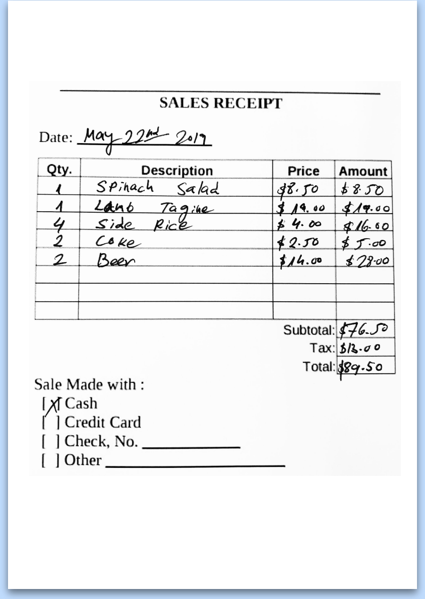


*With respect to size of computation

CI STATEMENT
total=$89.50

PROVER
Party producing proof
(Grocer)

VERIFIER
Party checking proof
(Customer)



SALES RECEIPT

Date: May 22nd 2019

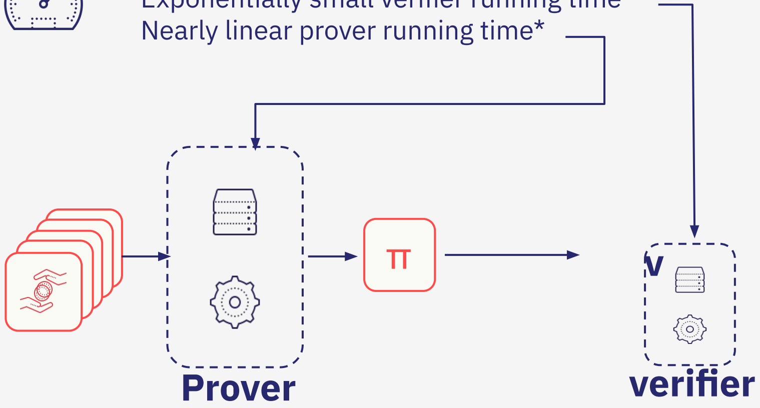
Qty.	Description	Price	Amount
1	Spinach Salad	$8.50	$8.50
1	Lamb Tagine	$19.00	$19.00
4	Side Rice	$4.00	$16.00
2	Coke	$2.50	$5.00
2	Beer	$14.00	$28.00
		Subtotal:	$76.50
		Tax:	$13.00
		Total:	$89.50

Sale Made with :
☒ Cash
☐ Credit Card
☐ Check, No. ___________
☐ Other ___________________

ZK-STARK Proofs

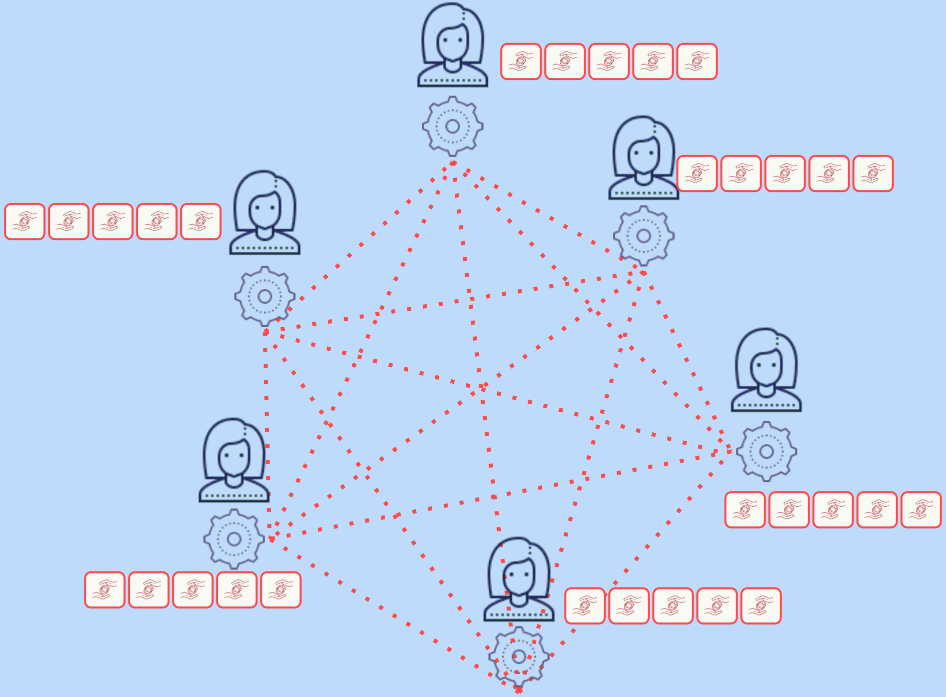
Privacy (Zero Knowledge, ZK)
Prover's private inputs are shielded

Scalability
Exponentially small verifier running time*
Nearly linear prover running time*



*With respect to size of computation

STARKWARE

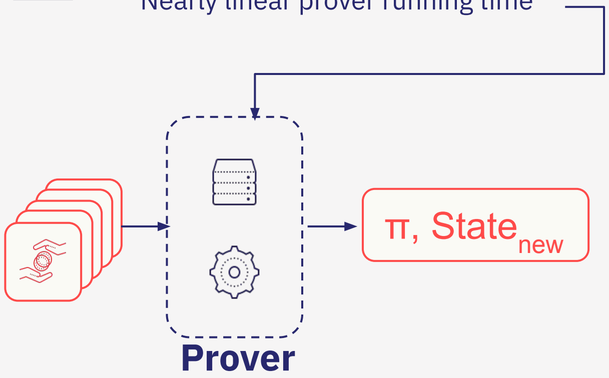


Verify (all transactions), don't trust

ZK-STARK Proofs

Privacy (Zero Knowledge, ZK)
Prover’s private inputs are shielded

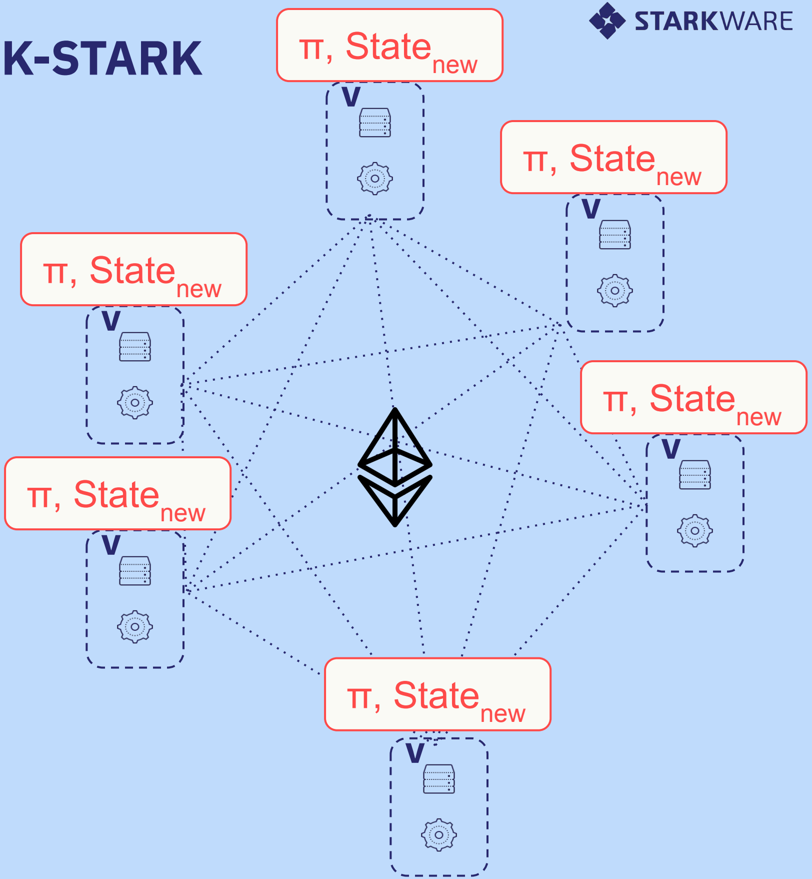
Scalability
Exponentially small verifier running time*
Nearly linear prover running time*



*With respect to size of computation

ZK-STARK

STARKWARE



Verify STARK proof, don’t trust

Two L2 Offerings

STARKEx

Largest L2 by TPS

Roughly same rate as Ethereum, rising

STARKNET

Alpha MainNet Launch
November 2021 !

StarkWare

Pedigree

Invented ZK-STARK, FRI, Cairo, SHARP, Validium, Volition, ...

Mission

Bringing scalability & privacy to a blockchain near you

Products

StarkEx Scalability Engine
StarkNet STARK-Rollup

STARKNET STARKEx

70

Team members

$160M

Funding (equity + EF grant)

STARKEx

As of February 17, 2022

Launched - June 2020

$420B
Cumulative Trading

106M
Tx Settled

>100K
Registered Users

36M
NFTs Minted

600K
NFT Mints/Proof

486
Gas/tx

dYdX

sorare

IMMUTABLE

DeversiFi

StarkNet

Decentralized Permissionless Validity-Rollup
offering scalable & secure Ethereum-like state

L2

SMART CONTRACTS

GENERAL COMPUTATION

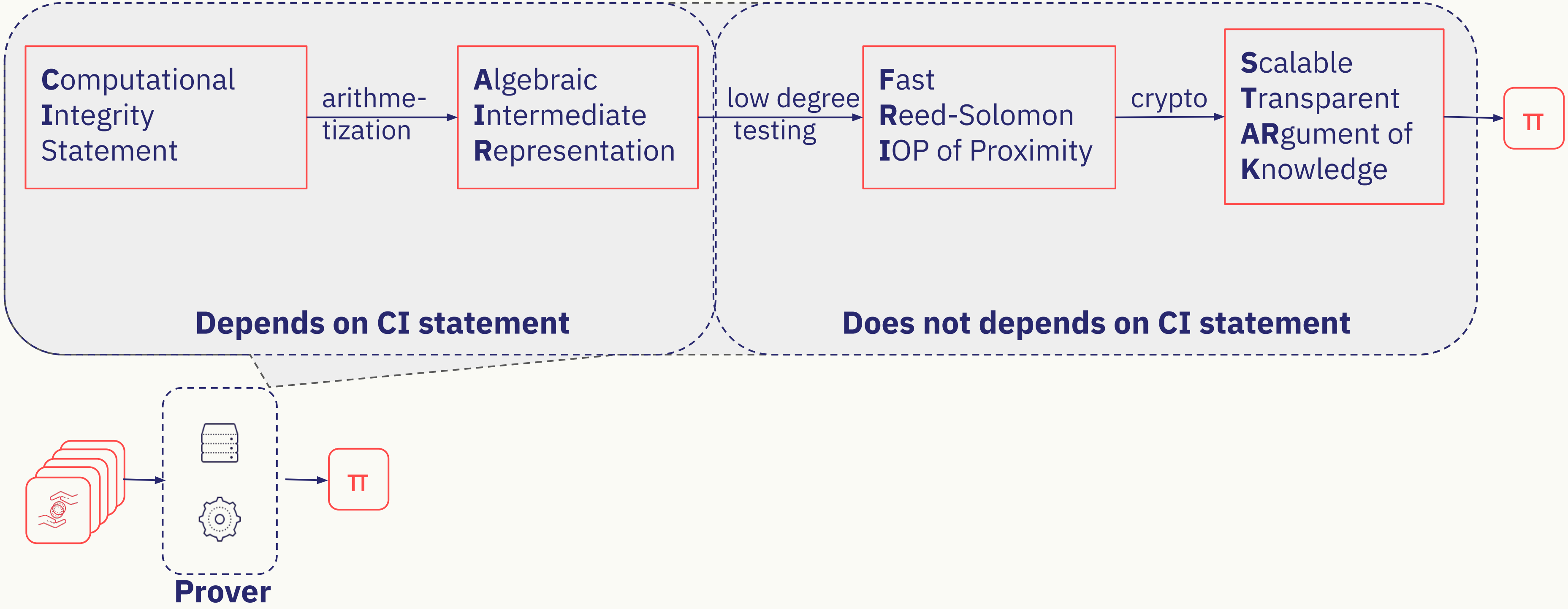
COMPOSABILITY

StarkNet Resources

- Learn
  - StarkNet/Cairo 101
  - Hello StarkNet!
- Explore the Ecosystem
  - StarkNet.io
  - Awesome StarkNet
- Stay up to date
  - StarkNet roadmap
  - StarkNet unofficial newsletter

How to build a STARK?

How to build an AIR-FRI STARK



Computational **I**ntegrity **S**tatement

arithme-tization

Algebraic **I**ntermediate **R**epresentation

low degree testing

Fast **R**eed-Solomon **I**OP of Proximity

crypto

Scalable **T**ransparent **AR**gument of **K**nowledge

π

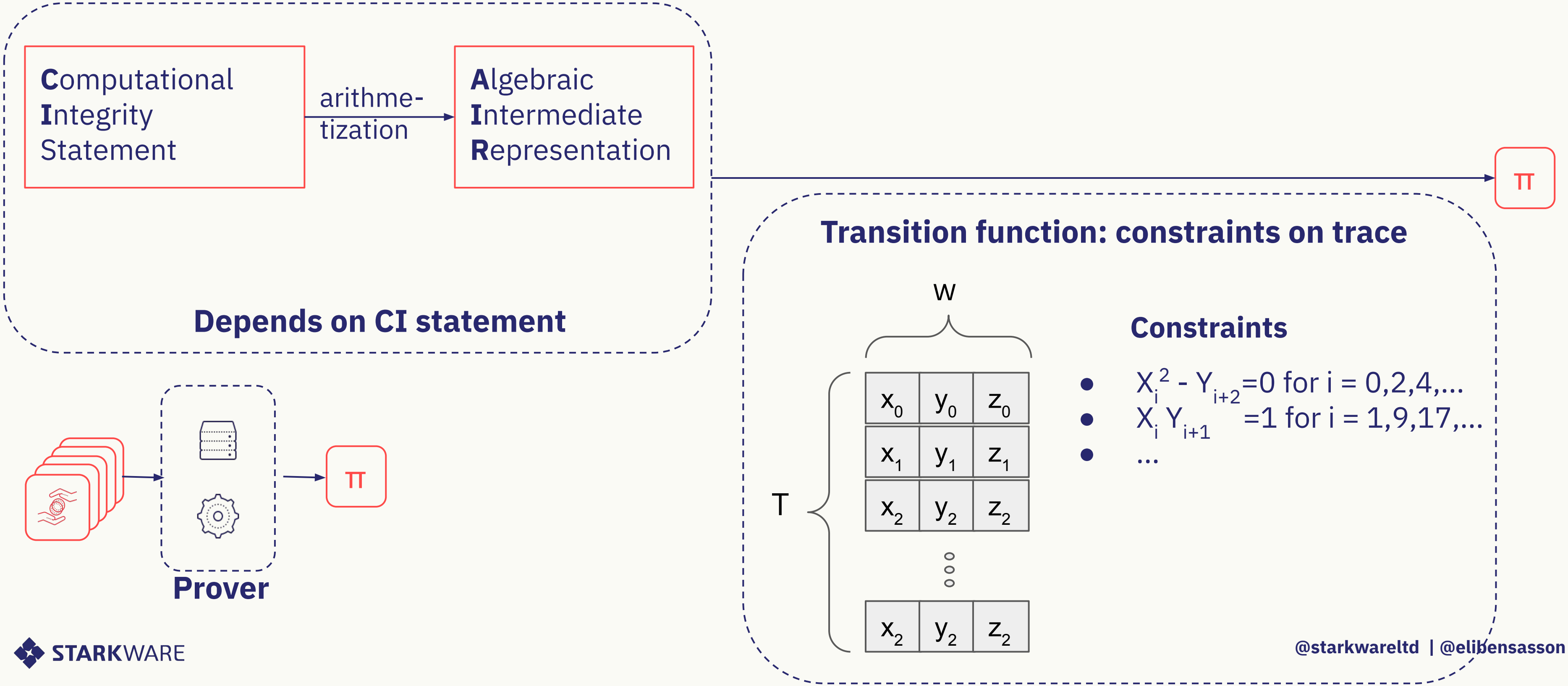
Depends on CI statement

Does not depends on CI statement

π

Prover

How to build an AIR-FRI STARK



Computational **I**ntegrity Statement

Algebraic **I**ntermediate **R**epresentation

Depends on CI statement

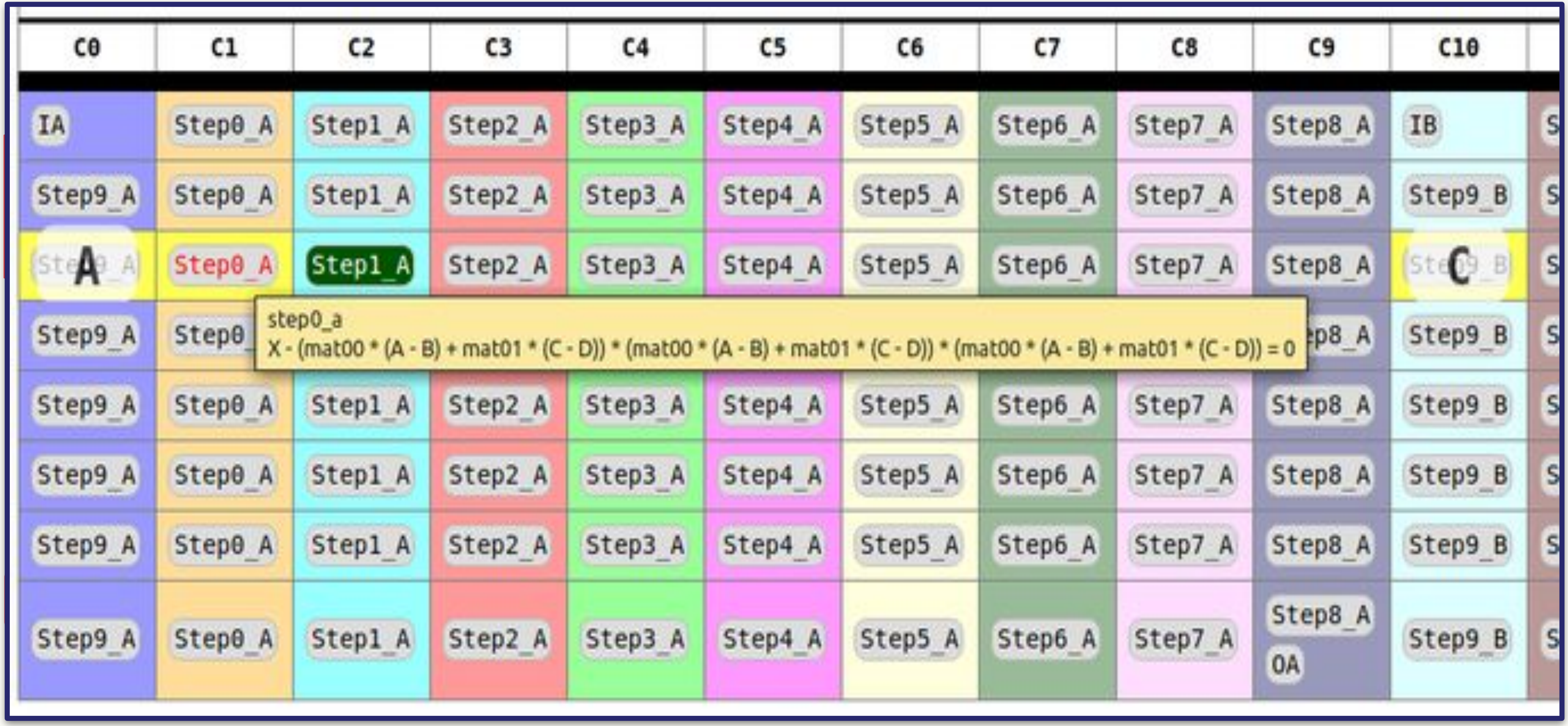
Transition function: constraints on trace

x_0	y_0	z_0
x_1	y_1	z_1
x_2	y_2	z_2
	⋮	
x_2	y_2	z_2

Constraints

- $X_i^2 - Y_{i+2}$=0 for i = 0,2,4,...
- $X_i Y_{i+1}$ =1 for i = 1,9,17,...
- ...

AIR Visualizer

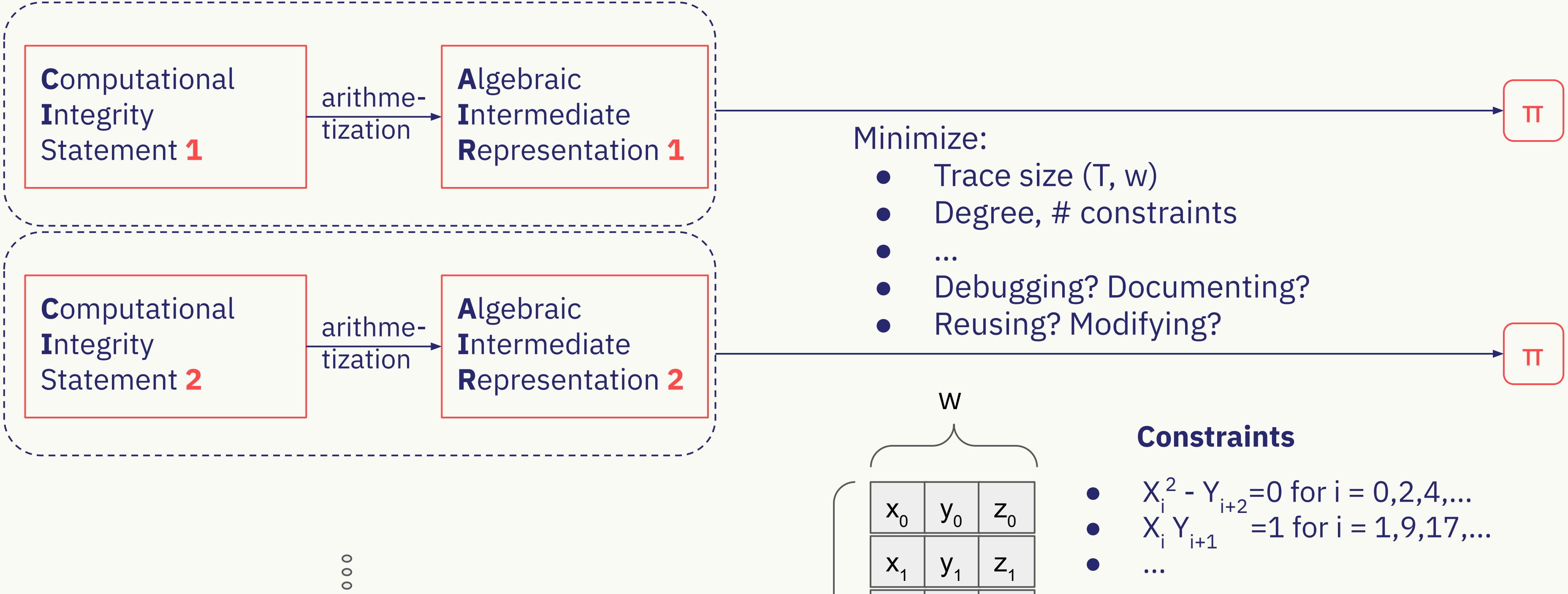


C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	
IA	Step0_A	Step1_A	Step2_A	Step3_A	Step4_A	Step5_A	Step6_A	Step7_A	Step8_A	IB	S
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4_A	Step5_A	Step6_A	Step7_A	Step8_A	Step9_B	S
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4_A	Step5_A	Step6_A	Step7_A	Step8_A	Step9_B	S
Step9_A	Step0								p8_A	Step9_B	S
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4_A	Step5_A	Step6_A	Step7_A	Step8_A	Step9_B	S
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4_A	Step5_A	Step6_A	Step7_A	Step8_A	Step9_B	S
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4_A	Step5_A	Step6_A	Step7_A	Step8_A	Step9_B	S
Step9_A	Step0_A	Step1_A	Step2_A	Step3_A	Step4_A	Step5_A	Step6_A	Step7_A	Step8_A OA	Step9_B	S

step0_a

X - (mat00 * (A - B) + mat01 * (C - D)) * (mat00 * (A - B) + mat01 * (C - D)) * (mat00 * (A - B) + mat01 * (C - D)) = 0

ASIC-like STARK

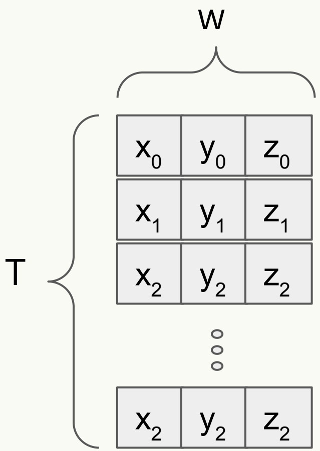


Computational **I**ntegrity Statement **1** $\xrightarrow{\text{arithme-tization}}$ **A**lgebraic **I**ntermediate **R**epresentation **1** → π

Computational **I**ntegrity Statement **2** $\xrightarrow{\text{arithme-tization}}$ **A**lgebraic **I**ntermediate **R**epresentation **2** → π

Minimize:

- Trace size (T, w)
- Degree, # constraints
- …
- Debugging? Documenting?
- Reusing? Modifying?



x_0	y_0	z_0
x_1	y_1	z_1
x_2	y_2	z_2
⋮	⋮	⋮
x_2	y_2	z_2

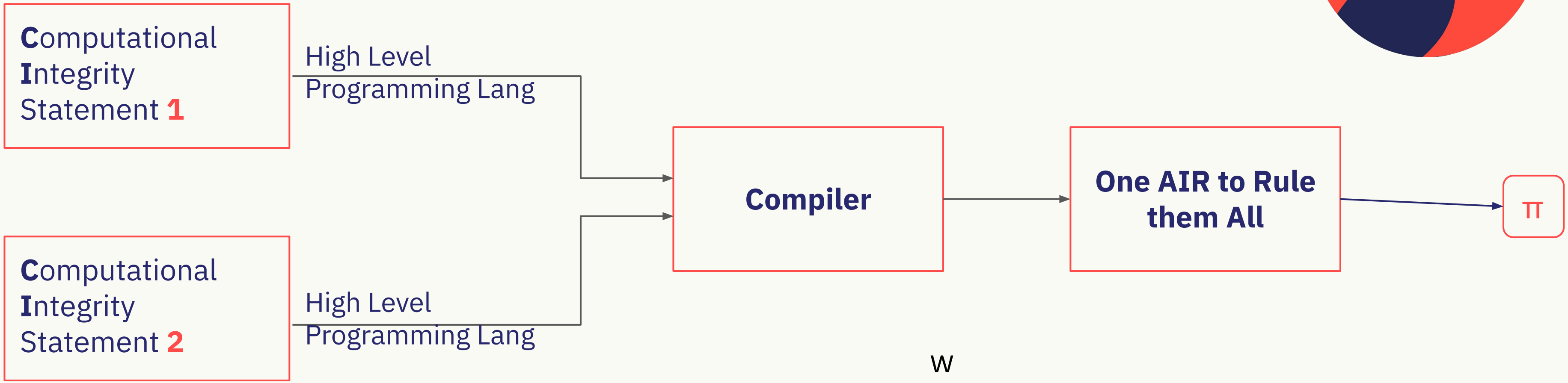
Constraints

- $X_i^2 - Y_{i+2} = 0$ for i = 0,2,4,…
- $X_i\,Y_{i+1} = 1$ for i = 1,9,17,…
- …

STARKWARE

@starkwareltd | @elibensasson

CPU AIR - CAIRo



Computational **I**ntegrity Statement **1**

High Level Programming Lang

Computational **I**ntegrity Statement **2**

High Level Programming Lang

⋮

Compiler

One AIR to Rule them All

π

$$T\left\{\overbrace{\begin{array}{|c|c|c|}\hline x_0 & y_0 & z_0\\ \hline x_1 & y_1 & z_1\\ \hline x_2 & y_2 & z_2\\ \hline & \vdots & \\ \hline x_2 & y_2 & z_2\\ \hline\end{array}}^{w}\right.$$

One AIR to Rule Them All

$w < 50$
# constraints < 100
Degree = 2
Variable T (depends on prog)

Cairo Theory

Cairo is 1st

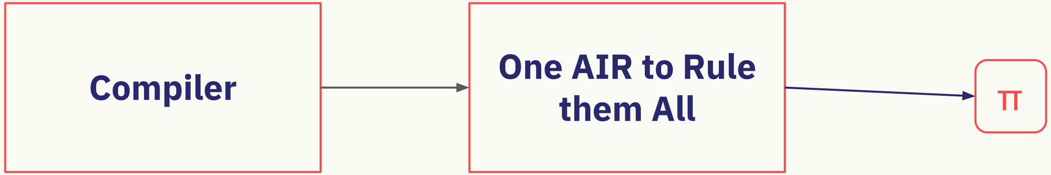
- **Universal** Von Neumann **ST**ARK

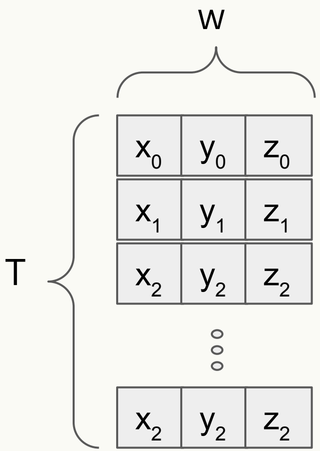
Scalability
Exponentially small verifier running time*
Nearly linear prover running time*

Transparency
No toxic waste (i.e. no trusted setup)

Universality
Applicability to general computation

- Universal Von Neumann verifier on blockchain (Ethereum Mainnet)





x_0	y_0	z_0
x_1	y_1	z_1
x_2	y_2	z_2
⋮		
x_2	y_2	z_2

One AIR to Rule Them All

$w < 50$
# constraints < 100
Degree = 2
Variable T (depends on prog)

Cairo Theory

Cairo is 1st

- **Universal** Von Neumann **ST**ARK

Scalability
Exponentially small verifier running time*
Nearly linear prover running time*

Transparency
No toxic waste (i.e. no trusted setup)

Universality
Applicability to general computation

- Universal Von Neumann verifier on blockchain (Ethereum Mainnet)

Resources:

Cairo landing page: https://cairo-lang.org/

Cairo whitepaper: https://www.cairo-lang.org/cairo-whitepaper/

Automated Theorem proving of Cairo soundness: https://arxiv.org/abs/2109.14534

Overview

1. My story and “red pill” moment
2. The Cambrian Explosion of ZKPs
3. ZK-STARKs unleashed
4. How to build a STARK?
5. [Fast RS IOPPs (FRI)] *time permitting*
    a. *STARK 101 online course: https://starkware.co/stark-101/*
    b. *STARK Math primer and whitepapers: https://starkware.co/stark/*

STARKWARE

Questions?

Eli Ben-Sasson / Co-Founder & President

@elibensasson | @starkwareltd

November 2021